International Journal of Scientific Research in Multidisciplinary Studies
Vol.7, Issue.6, pp.28-35, June (2021)

Research Paper
E-ISSN: 2454-9312
P-ISSN: 2454-6143

Voice Driven Bot For Cross Domain Database Querying

Affra H.[1*], Nageshwari S.[2], Poorvika A.N.[3]

[1,2,3]Information Science and Technology, College of Engineering Guindy, Anna University, Chennai, India

[*]*Corresponding Author: rgeetha@yahoo.com*

Available online at: www.isroset.org

Received: 08/Jun/2021, Accepted: 18/Jun/2021, Online: 30/Jun/2021

***Abstract*—** Many applications in health, medicine, finance store their information in relational database. Users cannot precisely work with structured query languages like SQL unless they have strong prior knowledge in this field. Also SQL is difficult to expertise for non-technical users. Hence a long-standing goal is to allow the users to interact with database with natural language. The reason for Voice-Driven bot is it can also be an assistive technology for visually impaired people.

The system involves 3 main phases : Speech to text, Text to SQL queries and Text to speech. In the First phase,the input is received as voice signals which will be used to predict the text from the audio file. Then in the second phase, SQL queries are generated from the text using encoder- decoder mechanism and pick the one which is valid and less complex to fetch the results. The model is trained using the Spyder dataset which makes the model aware of relations between the tables. After that, the results are converted to complete sentence and delivered back as a voice reply to the user. The natural language query from the user is converted to text using Speech recognition. Deep Learning is used to train the neural networks on large scale data of questions and answers.Bridge model finds the table names, column names and the conditional operators. SQLite Database is used to fetch the results based on the generated query

***Keywords**: Speech Recognition, SQL, BRIDGE model, NLP*

1. INTRODUCTION

The objective of the system is to design a bot which answers the user query by converting the natural language into SQL query automatically. Speech Recognition system and natural language processing have given rise to powerful voice-based interfaces. The system consists of three modules: Speech to text, Text to SQL queries and Text to speech. The first module is to convert the user query from speech to text. It is done using the Mozilla DeepSpeech model. In the following module the text is used to generate the SQL query using the encoder-decoder mechanism. And the last module is to convert the text back to speech. Here, the results are fetched from SQLlite database and google API is used for converting the end results to voice.

The report is organised as follows: Section 1 contains Introduction which gives an overall outline about the NLP based query retrieval, voice bot based solutions and objective of the model, Section 2 contains the related work carried before the beginning of the work, Section 3 provides the diagrammatic representation and the functionalities of each and every module is clearly explained, section 4 gives the detailed design level aspect of the entire model and the algorithm of each module and the training process and testing, Section 5 describes how the entire system is implemented and is used to analyse the certainty of success and failure with the help of the result obtained and a deep analysis on the performance of the given model is done, Section 6 concludes the overall and future works.

2. RELATED WORK

Parsing Natural language query to fetch results involves the task of translating the NL utterance into text. The conversion of speech into text and NLP based database querying techniques are widely explored in this section.

2.1 End-to-End Speech Recognition using RNN

The paper by Dario Amodei et al., [3] discussed the Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. Here, Recurrent neural network trained to ingest speech spectrograms and generate text transcriptions. This method involves 11 layers including many bidirectional recurrent layers and convolutional layers. CTC loss function was used to train the model. A very high ¬performing recognizers are created for two very different languages, English and Mandarin, required essentially no expert knowledge of the languages. But training involves very high end system which will be hindrance when one try to develop the model for their native languages.

2.2 Speech Recognition System

The paper by Pranab Das et al., [7] provides a Speech Recognition System which converts a voice into a text. This system consists of five modules. They are, Receiving

Voice Input, Sampling, Vectorization, Acoustic Model and Predicting words by Language Model. First, the received voice from the microphone will be in the form of one dimensional waves. To turn the received waves into number, the height of the wave at the equally spaced points are recorded. Then the input to the decoder will be 20 millisecond audio chucks. It will try to figure out the letters corresponding to the input. Having memory of the previous predictions, the valid words are identified. It will reduce the data loss by imposing weighted value to the data regions. So that the data which are about to loss can be reduced .The drawback here is this model makes use of MFCC algorithm. Since MFCC low robust to Noise signals, all inputs are altered by noise signal, if at least one frequency is distorted.

2.3 BERT based model for SQL generation

The paper by Tong Guo et al., [10] gives a detailed explanation about a simple methods BERT ¬based model for solving text ¬to ¬SQL problem. It is observed that some of the table content matches some words in question string and some table header may match some words in question string. Here, encoding of two addition feature vector is carried for the deep model. It is trained using the mask language model loss and the next¬ sentence loss. And then it is fine¬ tuned for specific tasks like text classification, text matching and natural language inference. Here, BERT as the representation layer. It gives very high accuracy on WikiSQL dataset which includes only simple queries. It fails to create querieswhich includes multiple tables.

2.4 Syntax Tree based model for SQL generation

The paper by Tao Yu, et al., [9] provides a Syntax Tree based model for Natural language processing-based database querying. Using syntax tree-based model, generating a complex SQL query with multiple clauses, selections and sub queries are efficiently possible. In this model, each SQL token is predicted with the help of grammar rules and SQL generation path history. Totally nine modules are existing in SQL token prediction. They are, IUEN module, KW module, COL module, OP module, AGG module, Root/Terminal module, AND/OR module, DESC/ASC/LIMIT module and HAVING module. The self-developed module working strategy says that If the SQL generation path history's last prediction is HAVING clause, then there is a chance for aggregate functions (SUM(), COUNT(), AVG(), MIN(), MAX(),..) takes place next. As a result, it achieves 12.3 percentage total improvement compared to previous models and 22 percentage accuracy. The model will degrade the correctness of predicted SQL query since the context of the input is not taken into an account.

2.5 Bridging Textual and Tabular Data In cross Domain semantic parsing for SQL generation

The paper by Xi Victoria Lin et al., [11] explained the Bridging Textual and Tabular Data for Cross Domain Text to SQL Semantic Parsing. Here, Question Schema Serialization and Encoding is carried out for Metadata Features extraction which is done using BERT and LSTM. Then Bridging is carried out which acts as a anchor text to link value mentions in the question with the corresponding DB fields. Finally Schema Consistency Guided Decoding is carried out where SQL query is generated based on the SQL syntax constraints. It fails to give compositional generalization and the application of BRIDGE in other tasks are yet to be identified. It gives the accuracy of 71% on Spyder dataset.

2.6 Dependency Graph for SQL Generation

The paper by Xiaojun Xu et al., [12] discussed synthesized SQL queries from natural language without the use of reinforcement learning. In this, sketch¬ based approach is employed where sketch contains a dependency graph so that one prediction depends only on the previous predictions not on the entire one. To denerate the where class, sequence¬ to ¬set and column attention mechanism is employed. The former is to predict an unordered set of constraints and the later is to capture the dependency relationship. The main aim is to avoid sequence¬ -to --sequence model where SQL query order does not matter. This achieves the accuracy of 74.1% when tested with Wiki SQL dataset. But it is noticed that this approch is not effective when the query gets complex and it involves many table and joins.

3. SYSTEM DESIGN

The system consists of three modules. The first module is to convert the user query from speech to text. It is done using the Mozilla DeepSpeech model. In the following module the text is used to generate the SQL query using the encoder-decoder mechanism. And the last module is to convert the text back to speech. Here, the results are fetched from SQLlite database and google API is used for converting the end results to voice.

3.1 SPEECH TO TEXT MODULE

The natural language query spoken by the user is received as put for the module and stored in the audio file. For processing them to text, it requires the following stages. They are Sampling, Vectorization, Decoder using Acoustic and Language Model.

3.1.1 Dataset for Speech Recognition:

The deepspeech model is trained with 3816 hours of transcribed audio. The model also includes around 1700 hours of transcribed WAMU (NPR) radio shows. For further training, the dataset is created by recording the Natural Language query from users and storing them as audio files. 300 audio files along with their ground truths are used for training.

3.1.2 Sampling

Initially we perform sampling on the audio files where continuous signal is converted to discrete signals using the heights of the signal at the equally spaced points. Here sample rate, bits per sample and channel number are considered.

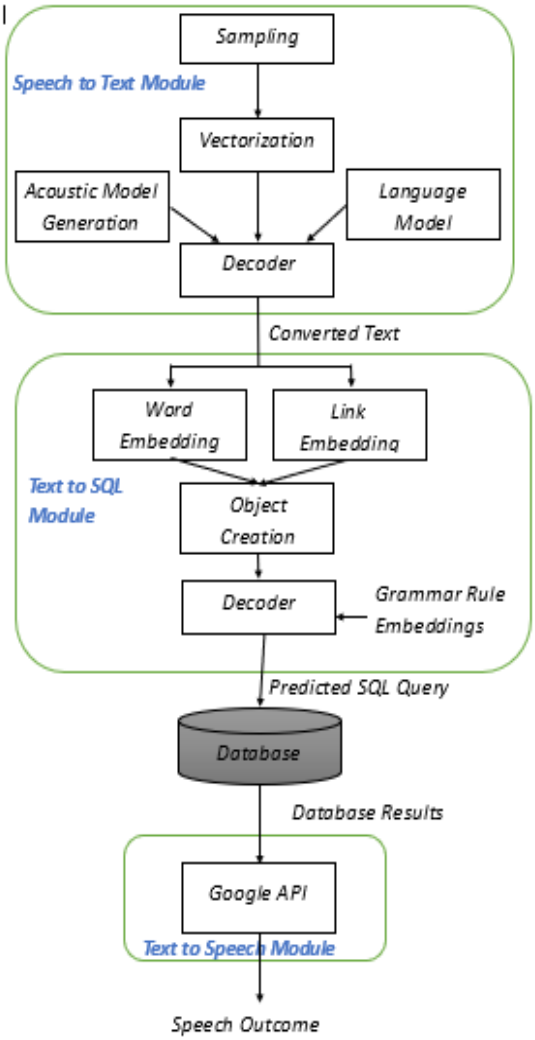

Figure 1 Architecture Diagram

3.1.3 Vectorization

Feature extraction is done using Mel Frequency Cepstral Coefficients (MFCC). Vectorization is performed on the sampling output and a NumPy buffer array is obtained. At this stage only the useful signals will be retained since redundant and unwanted information is eliminated using the blocking mechanism. This array will be passed as a input to the decoder.

3.1.4 Decoder

Decoding is carried out with the help of acoustic and language model. Acoustic model contains statistical representations of each distinct sounds that makes up a word. It is a RNN model which involves 5 hidden layers with 3 non recurrent layers, 1 recurrent layer and 1 output layer. Language model provides the context to distinguish the sounds or the word that sounds similar by performing probability distributions. By means of probability distributions it will assign probability to a sequence of words. Therefore, with the help of acoustic model and language model the decoder will predict the appropriate text. The NL query in the text format corresponding to the audio given is converted to text using the generated model.

3.2 TEXT TO SQL MODULE

The user query in text format obtained from the speech-to-text module is processed in Word Embedding, Link Embedding, Object Creation and Decoder to generate the SQL query.

3.2.1 Dataset for Query Classification

The training of this module is carried out using the Spyder dataset which contains 10,181 questions and 5,693 unique complex SQL queries involving 200 databases with multiple tables across 138 different domains.

3.2.1 Word Embedding

The text obtained from previous module is cleaned for further processing. Stop words are removed and tokenized. Word embedding helps to identify the aggregate function. If the query contains aggregate function, it further predicts the type of aggregate function with the help of classification models. Classification model is trained using the binary table which includes features as columns and NL queries as rows. After predicting the type of query link embedding was performed.

3.2.2 Link Embedding

Link embedding helps to identify level of complexity by identifying the number of tables involved to generate query. Classification model is trained using the binary table which includes features as columns and NL queries as rows. After predicting the type of query link embedding was performed.

3.2.3 Object Creation using Bridge Model

Then Bridge model gives the raw information regarding the table and column names and it identifies the conditional operators with the help of database schema. In Bridge model, Encoding is carried out for Meta-data Feature's extraction which is done using BERT and LSTM. Then Bridging is carried out which acts as a anchor text to link value mentions in the question with the corresponding DB fields.

3.2.4 Decoder

Using the output of Bridge model, decoder conditionally append the SQL tokens using the grammar rule to generate the query in the proper format. The SQL query generated from the NL query will be the output of this module.

3.3 TEXT TO SPEECH MODULE

This module converts the results fetched from the database into the speech format which will be the final output of the model.

3.3.1 Database Creation

The obtained SQL query is used to fetch the results from the SQLlite database. Database is created with sample entries which is used to fetch results. The results are fetched in the tuple format. They are converted to string to perform speech conversion.

Google Text to Speech API commonly known as the gTTS API is used to convert the resultant string into speech format. The wave file of the NL query result obtained from gTTS API is the final output of the system.

4. METHODOLOGY

The detailed design of the model and algorithm of various modules are given in this chapter.

4.1 SPEECH TO TEXT USING DEEPSPEECH MODEL

Transcribing to text from the speech input provided by the user is explained in this section.

4.1.1 Sampling

The audio file containing NL queries, obtained from the user will be in the mp3 format. Then the audio will be sampled with the sampling rate of 16kHz. The bit rate is 16 so that 16 bits will be used to store the values of each sample. The number of channels is set as mono. Parameters are tuned using PyDub library.

Algorithm 1: Sampling of NL queries

Step 1: Convert the audio file to .mp3 format.
Step 2: Set sampling rate to 16kHz
Step 3: Set bits/sample to 16 bits
Step 4: Set the channel number to 1
Step 5: Measure the height of signal and convert the speech to discrete value
Step 6: Store the results in .wav format

4.1.2 Training the pre-trained model

Acoustic Model already developed by Mozilla is used to predict the phonetics. It is trained with around 68 hours of audio along with their transcripts. It mostly includes American English which doesn't give the desired accuracy for the Indian users. To improve the accuracy the model is further trained with the Indian English audio files.

Audio files are obtained from different users and stored in a folder. Sampling is performed for all audio files. CSV files are created which includes 3 columns. They are location of audio files, size of the file and ground truth. For training, validation and testing, the data-set is split into three CSV files as train.csv, dev.csv, test.csv.

Algorithm 2: Creation of dataset

Step 1: Collect the voice input from different users
Step 2: Store all audios in single folder
Step 3: Find the size of the audio files
Step 4: Create the CSV file which have location of audio files, size of audio files and ground truth of audio files as columns.
Step 5: Split the CSV into train,dev and test

Acoustic model is trained using the deep earning mechanism which is composed of 5 layers of hidden units.The first three layers are not recurrent. At each time step, the non-recurrent layers work on independent data. The fourth layer is a recurrent layer with two sets of hidden units. One set has forward recurrence while the other has backward recurrence. The fifth (non-recurrent) layer takes the forward units as inputs. The output layer predicts the character probabilities for each time step.

Training of acoustic model is initialized by providing the location of the checkpoints of the pre-trained model so that the training is continued on the already trained model of Mozilla. Further the location of binary files of the language model, train.csv, dev.csv and test.csv is given learning rate is set to 0.0001 which seems to give the better accuracy. After training, the model is stored in .pb format. It is converted to .pbmm format which is memory mapping format to predict the user inputs with great efficiency.

Algorithm 3: Training the pre-trained Model

Step 1: Perform sampling on all audio files
Step 2: Perform vectorization for feature extraction using MFCC.
Step 3: Provide the location of checkpoints of acoustic and language model.
Step 4: Train the pre-trained model with learning rate of 0.0001.
Step 5: After training, store the model in .pbmm format

The model obtained after training is loaded and predictions are made. The buffer array type is set as type16 since the bit rate during sampling is 16 bits. It is tested with different audio files. To know the better accuracy and loss of the trained model it is tested only with NL queries. Audio from the user is sampled and stored in .wav format.The wave file is then read to create a NumPy buffer array. The buffer is a byte array since pre-trained model expects 16-bit int array. Model is created with trained acoustic and language model along with the parameters. The NumPy buffer is given to the decoder which will predict the end result with the help of acoustic and language model.

4.2 SPEECH TO TEXT USING EXISTING MODEL

Natural language questions alone is extracted from the Spyder dataset and converted them to CSV format. CSV file is given as input to PYTTSX (offline Python Text to Speech library) to get output as speech format (Audio file) of NL questions. The output audio file is considered as source file to measure the performance of various API's. These API's includes Google Speech API, Pocketsphinx API, Wit.ai API. The created audio file is passed as input into these API's for Speech Recognition and the results are stored in separate CSV file. The csv file obtained is used to calculate the performance measures.

Algorithm 4: Performance measure of existing API's

Step 1: Extract NL queries from Spyder dataset and store in CSV file
Step 2: Convert NL queries to audio files using PYTTSX
Step 3: Convert audio files to text in 3 different API's
Step 4: Store the results in CSV and compare the results with transcripts
Step 5: Calculate the accuracy of each API

4.3 CLASSIFICATION OF NL QUERIES

Classification of natural language query to identify the type of query, aggregate functions and complexity. SQL queries obtained from spyder dataset is used to build the binary table. All the SQL queries with SUM(), COUNT(), AVG(), MIN(), MAX() are considered aggregate queries and the rest is considered simple queries. If the given query is identified as aggregate, they are further classified to identify the type of aggregate function. For that, Binary table is created with natural language query as rows and valuable tokens as columns. Then to identify the level of complexity, SQL queries obtained from spyder data-set is used to build the binary table. Queries involving single table and multiple tables are classified for identifying the tables using different classification algorithms. Several supervised learning algorithms are used to identify the best model of all.

4.3.1 Query Classification using Naïve Bayes

Naive Bayes is a classification technique based on Bayes Theorem with an assumption of independence among predictors. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes predicts membership probabilities for each class such as the probability that data point belongs to a particular class. It assumes that particular feature in a class is unrelated to the presence of any other feature. The class with the highest probability is considered as the most likely class. Then, Naive Bayesian equation is used to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

4.3.2 Query Classification using Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. SVM uses the data to obtain different points and maps them in a very high dimensional space using a nonlinear kernel function. SVM works by searching for the optimal solution for class splitting. The solution can be used to maximize the distance with respect to the nearest points. Finally, the hyperplane is obtained. For obtaining optimal results, parameters of SVM will be tuned.

4.3.3 Query Classification using Random Forest Tree

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random Forest initially select “k” features from total “m” features such as K<m. Among “k” features, calculate the node “d” using the best split point. This can be done with any decision tree algorithms. Later it will split the node into daughter modes using the best split based on the votes obtained. This is repeated until the specified number of nodes are reached.

To identify the best classification model and to attain the highest accuracy classification is carried out in all three. Further to improve the accuracy, feature selection is carried out where the features which doesn’t hold any values are removed and finally the cluster of 120 features are used.

Algorithm 5: Binary Table Creation

Step 1: Extract natural language query from Spyder dataset

Step 2: if Query contains ’min’, ’max’, ’avg’, ’sum’, ’count’ keywords then mark the classfield1 as ’aggregate’ otherwise mark as ’simple’

Step 3: if Query contains 'join','(select'),'(select' keywords then mark the classfield2 as ' complexTable' otherwise mark as 'singleTable'

Step 4: if Query contains aggregate function fill aggregate function is classfield3

Step 5: if token matches valid words, then fill the column as ’Y’ fill the column as ’N’

Step 6: Binary table created is loaded and transformed to binary value

Step 7: Models are generated using Gaussian Naive Bayes, SVM, Random Forest

Step 8: Classification model with highest accuracy is used to predict results

For identifying whether the query is Simple/Aggregate, Random Forest Tree gives the best results. For fine tuning the hyper parameters are modified as n_estimators=2000, criterion=’entropy’ and min_sample_split=10. This model gives the accuracy of 88%. For identifying the type of query, Support Vector Machine gives the highest accuracy of 91.6% when hyperparameter kernel=’rbf’. For identifying the query complexity, Support Vector Machine gives the highest accuracy of 71% with the hyperparameters kernel=’rbf’, gamma=1, degree=1

4.3 BRIDGE MODEL FOR OBJECT CREATION

Pre-trained Bridge model takes input as a natural language utterance and a database schema and generates SQL queries as token sequences. Here the meta data features are obtained using the BERT model where the tokens which helps to identify the table names, column names and conditional operators are identified. Later fussy string mapping is carried out to obtain the object which gives the raw information.

Algorithm 6: Creation of Bridge Model

Step 1: Create tokens for NL query and database schema

Step 2: Meta-data Features extraction using Question-Schema Serialization and Encoding

Step 3: Fussy string matching to map the NL tokens with column names and conditional operator

4.4 SQL GENERATION USING GRAMMAR RULES

Pre-processed Natural Language query and database schema are given to pretrained Bridge model which identifies the column names and ground truth values with accuracy of 86%. Text received from the previous module is used to perform text pre-processing, which consists of

removing punctuation, white spaces, duplication, tokenization and Stop-words removal. After obtaining column names and conditions from the BRIDGE model, they are conditionally appended with the help of grammar rules to generate the SQL query which will be syntactically correct.

Algorithm 7: Generation of SQL Query

Step 1: Pre trained Bridge model identifies table names, column names, conditional operators

Step 2: " SELECT " and column names are appended to query variable

Step 3: " FROM" and table names are appended later

Step 4: if conditional operators present then conditions are appended

Step 5: Generated query verified with classification results

4.5 TEXT-TO-SPEECH COMVERSION

Generated SQL query is used to fetch the results. Databases are created using SQLite, appropriate tables and sample entries are created. Results are fetched in the form of tuples. Results fetched are converted to string for voice conversion. Google API commonly known as gTTS is used for converting the SQL results in the form of voice to users.

Algorithm 8: Generation of Query result

Step 1: Creation of database using SQLite

Step 2: Sample entries for all tables and columns of single database are created

Step 3: Connection is created for the created database

Step 4: Generated SQL query is given to database

Step 5: Results obtained in tuples format are converted to string

Step 6: Results are given to google API

Step 7: Audio file in .wav format is obtained

5. RESULTS AND DISCUSSION

In this Study, Voice driven bot for cross domain database querying, The Prediction of speech to text is done using the pretrained Mozilla's DeepSpeech model with the accuracy of 75%. To improve the accuracy further while predicting Indian English, new dataset is created. For that, the natural language questions are extracted from spyder dataset which is around 300 entries and stored in CSV file. User queries from spyder dataset are shown in Table 5.1. They are converted to audio files using PYTTSX which is in .wav format. Training Dataset includes three columns. They are, Location of the audio files, Size of the audio files and Trascription of the audio files. Training dataset shown in Table 5.2.

Table 5.1 User Queries from Spyder Dataset

S.NO	NATURAL LANGUAGE QUERY
1	How many heads of the department are older than 56?
2	List the name, age and born state of the heads of the department order by age.
3	What are the maximum and minimum budgets of the department?
4	In which year are most departments established?
5	How many farms are there?

To predict the text from given speech, Dataset is converted to feature vector and given to recurrent neural network. It converts speech spectrograms to English text. Average loss of model when tested with multiple instances are recorded to be 16%.

Table 5.2 Training Dataset

WAV_FILENAME	WAV_FILESIZE	TRANSCRIPT
OUTPUT1.WAV	1058458	who scored more that eighty percent
OUTPUT2.WAV	1058458	how many are from it department
OUTPUT3.WAV	1058458	give me the total girls count
OUTPUT4.WAV	1058458	who secured the highest score
OUTPUT5.WAV	1058458	find the name of the students whose performance is good

To measure the performance of three different existing API, user queries are collected from the Spyder dataset which is around 100 entries and stored in CSV file. The extracted NL questions are converted to audio files using PYTTSX. This audio files are given to Google Speech to text API, Pocketsphinx API and Wit.ai API as an input. The result obtained from each API are analyzed. After analysis it is found that WIT.ai API has the highest accuracy as shown in table 5.3.

To convert Natural language questions to Structured Query Language, pretrained bridge model is used. It gives information regarding columns, tables and column values. Then SQL queries are predicted using this information.

Table 5.3 Performance measure of existing APIs

API NAME	ACCURACY
GOOGLE API	74.78%
POCKET SPHINX	84.50%
WIT.AI API	89.73%

To check the correctness of SQL query, Classifications are done using four different models. They are, Naive Bayes, Decision tree, Random Forest and Support Vector Machine. Binary tables are created to train the classification model. For that, Natural language questions extracted from spyder dataset which is around 10,000 entries are kept as rows. The extracted texts are pre-processed and tokens which helps to classify a query alone is separated which is around 120 and kept as columns. Those features are shown in table 5.7. The binary table for classification is shown in table 5.4, 5,5, 5.6. Word Embedding and Link Embedding are done to create a class labels which is a column of binary dataset.

Table 5.4 Binary table for Simple/Aggregate query classification model

NL Question	Class Label	How	Many	List
How many heads are older than 56?	Aggregate	Y	Y	N

List the name and age of the heads of departments.	Simple	N	N	Y
What are the minimum budget of the departments?	Aggregate	N	N	N

Table 5.5 Binary table for Aggregate query type classification model

NL Question	Class Label	How	Many	Average
How many heads are older than 56?	Count	Y	Y	N
What is the average number of working horses?	Average	N	N	N
What are the minimum budget of the departments?	Maximum	N	N	N

Table 5.6 Binary table for Single/multiple table classification model

NL Question	Class Label	How	Many	what
In which year were most departments established?	Single	N	N	N
What are the distinct ages of the heads who are acting?	Multiple	N	N	Y
How many farms are there?	Single	Y	Y	N

Table 5.7 Features of binary dataset

how, many, minimum, maximum, count, total, average, sum, who, what, where, least, in, most, order, sorted, bigger, smaller,	which, distinct, list, more, ascending, descending, give, each, show, reversed, greater, at, lower, less, never, when, longest,	greatest, ending, starting, furthest, top, bottom, beginning, select, increasing, decreasing, long, range, frequent, partition, contain,	max, min, higher, start, end, oldest newest, smallest, shortest, largest mean, lowest, lasted, ended, always, cheap, etc...

In Word Embedding, Queries are classified as Simple and Aggregate. Queries are labeled as Aggregate when it contains a words such as Min, Max, Sum, Avg and Count. Otherwise labeled as Simple. The results shows 5900 queries are simple queries and 3800 queries are aggregate queries. Decision Tree for Simple/Aggregate classification model is shown in Figure 5.1. Another classification made on Types of query. Here Aggregate Queries alone is separated and labeled by five different class labels. Here out of 3800 queries, 174 queries are labeled as Max,288 queries are labeled as sum, 2710 queries are labeled as count, 428 queries are labeled as avg and 200 queries are labeled as Min aggregate functions respectively.

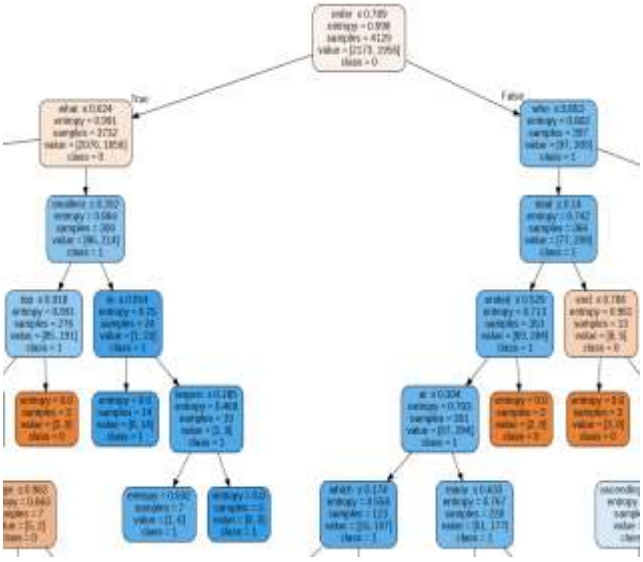

Figure 5.1 Decision Tree of Simple/Aggregate SQL Query Classification Model

In Link Embedding, tables associated with SQL queries are found. Here the queries are classified into two different labels. Queries contains the words like join, union, select and intersection are classified as multiple table query. Which means the query derived from more than one table. otherwise, queries classified as single table query. Here 4738 queries are labeled as Single Table queries and 4955 are labeled as Multiple Table queries.

Table 5.8 Accuracy of Classification model

Machine Learning Classifier	**Simple/ Aggregate Query**	**Type of Aggregate Function**	**Single/ Multiple Table Query**
Support Vector Machine	85%	92%	68%
Random forest	88%	91%	66%
Decision tree	83%	87%	60%
Decision tree with feature selection	85%	87.5%	63%
Decision tree with parameter tuning	86%	90%	65%
Naive Bayes	51%	25%	59%

Binary tables are given as an input to Four classification models. The accuracy obtained from each model is shown in table 5.8. The prediction results are shown in table 5.9.

Table 5.9 The prediction Model Results

Classification Model	**Actual SQL Query**	**Correctly Predicted SQL Query**
Simple/Aggregate	9693	8530
Type of Aggregate Function	3800	3500
Single/Multiple table	9693	6593

SQLite is used to create a database. Then tables and Sample entries are created. The generated SQL query is parsed to the Database to fetch the results. The results are converted to Voice format using Google Text to Speech API. The result is shown in Figure 5.2.

```
Upload the input voice and click enter: >
list the department names
Converted text : list the names of the department
Successfully Connected to SQLite
100% 1/1 [00:02<00:00,  2.68s/it]
SQL: SELECT department.Name FROM department
Type of query : simple query
Number of tables involved : Single table
('IT',)
('CSE',)
('ECE',)
<IPython.lib.display.Audio object>
Upload the input voice and click enter: >
```

Figure 5.2 Final Result

6. CONCLUSION AND FUTURE SCOPE

The aim of this system is to convert the natural language questions into SQL queries and produce a voice based results. This system involves three modules named 'Voice to Text', 'Text to SQL' and 'Text to Voice'. First, Voice to Text module (Converting a Speech to Text) has been done by using deep learning method with accuracy of 76%. Then the Performance Measurement of the existing API are carried out to examine which API is best among existing APIs. Three APIs named 'Google API', 'Wit.ai API' and 'sphinx API' was taken. These are examined by converting batch of Voice to Text. The result shows Wit.ai API is best Bridge model is used to create an object which is used to convert a Text into a SQL query with the help of Grammar rules. Supervised Machine Learning algorithms like Naive Bayes, Decision Tree, Random Forest and Support Vector Machine are used for Query classification. For generated SQL query Results are fetched from SQLite database. The results are converted to voice using Google API.

The future work of this system will include, improving prediction of Text from different Voice tones and Improving accuracy in conversion of speech to text. Since the training is carried out only limited audio files. Accuracy seems low when the input involves more complex queries since the accuracy of our model

REFERENCES

[1] Ahmed Elgohary, Saghar Hosseini, Ahmed Hassan Awadallah, "Speak to your Parser: Interactive Text-to-SQL with Natural Language Feedback", arXiv:2005.02539v2 ,**2020.**

[2] Bailin Wang, Richard Shin,Xiaodong Liu, Oleksandr Polozov, Matthew Richardson, "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers", In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, **2020.**

[3] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, Zhenyao Zhu , "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin", **2015.**

[4] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, Dong Ryeol Shin, "RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases", Association for Computational Linguistics, **2020.**

[5] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, Dongmei Zhang "Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation", **2019.**

[6] Jichuan Zeng, Xi Victoria Lin, Caiming Xiong, Richard Socher, Michael R. Lyu, Irwin King, Steven C.H. Hoi, "Photon: A Robust Cross-Domain Text-to-SQL System", **2020.**

[7] Prerana Das, Kakali Acharjee, Pranab Das and Vijay Prasad, "Voice Recognition System : SPEECH-TO-TEXT", International Journal of Applied and Fundamental Sciences,**pages 191-195, 2016.**

[8] Puneet Kaur, Bhupender Singh,Neha Kapur, "Speech Recognition with Hidden Markov Model", **2014.**

[9] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev, "Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task", In Proceedings of the Conference on Empirical Methods in Natural Language Processing pages **1653–1663,2018.**

[10] Tong Guo ,Huilin Gao "Content Enhanced BERT-based Text-to-SQL Generation ",arXiv:1910.07179v5 , **2020.**

[11] Xi Victoria, Lin Richard, Socher Caiming Xiong,"BRIDGE : Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing", **2020.**

[12] Xiaojun Xu, Chang Liu, Dawn Song " SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning", **2017.**

