

QUERY GENERATION USING NADAQ SYSTEM

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in
Computer Science and Engineering

by

Bhumika Reddy kothapalli, Pradeepthi Kavuri (38110267, 38110245)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL - 2020



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Bhumika Reddy kothapalli, Pradeepthi Kavuri (38110267, 38110245)**, who carried out the project entitled "**Query Generation using NADAQSystem**" under my supervision from November 2019 to April 2020.

Internal Guide

Dr. Senduru. Srinivasulu, M.E., Ph.D

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D., and Dr. L. LAKSHMANAN M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We **Bhumika Reddy kothapalli, Pradeepthi Kavuri (38110267, 38110245)** hereby declare that the Project Report entitled “**Query Generation using NADAQ System**” done by me under the guidance of **Dr. Senduru. Srinivasulu, M.E., Ph.D**

(Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing , Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D., Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.Senduru.Srinivasulu, M.E., Ph.D** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the Completion of project.

ABSTRACT

Natural language processing (NLP), as a theory motivated computational technique, has extensive applications. Automated test case generation based on path coverage, which is a popular structural testing activity, can automatically reveal logic defects that exist in NLP programs and can save testing consumption. The high complexity behind SQL language and database schemas has made database querying a challenging task to human programmers. In this paper, we present our new natural language based database querying system as an alternative solution, by designing new translation models smoothly fusing deep learning and traditional database parsing techniques. We develop new techniques to enable the augmented neural network to reject queries irrelevant to the contents of the target database and recommend candidate queries reversely transformed into natural language.

TABLE OF CONTENTSS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	VIII
1	INTRODUCTION	1
	MACHINE LEARNING	1
	SUPERVISED LEARNING	2
	Regression	4
	Classification	4
	EXISTING SYSTEM	4
	PROPOSED SYSTEM	5
2	LITERATURE SURVEY	6
3	METHODOLOGY	10
	SYSTEM PROCESS	10
	SYSTEM DESIGN	11
	Loading the Live Data	11
	Data Pre-Processing	12
	Building the Model	13
	Training The Model	13
	SOFTWARE DEVELOPMENT	15
	Waterfall Model	16
	RAD Model	17
	LANGUAGE DESCRIPTION	20
	Python Line Structure	21
	Python Data Types	23
	Selection of Integrated Development Environment	28
	Selection of Operating System	30
	Selection of Web Application Framework	30
4	RESULTS AND DISCUSSION	32
5	CONCLUSION AND FUTURE WORK	35
	CONCLUSION	35
	FUTURE WORK	35

REFERENCES	37
APPENDIX	38
[A] SOURCE CODE	38
[B] SCREEN SHOTS	57
[C] PAPER	58

LIST OF FIGURES

Figure No.	Title	Page No.
1.1	Supervised prediction model	3
3.1	System Architecture	10
3.2	Waterfall Project Management Methodology	16
3.3	Rapid Application Development Methodology	17
3.4	Python Supporting Operating Systems	20
4.1	Classroom table result	32
4.2	Professor table result	33
B.1	Home Page	57
B.2	Text Input Page	57
B.3	Creating Database	58
B.4	Database Input	58
B.5	List of Roles	59

CHAPTER 1

INTRODUCTION

MACHINE LEARNING

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future. Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms. However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result. The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms, with an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it.

These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

SUPERVISED LEARNING

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism’s genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x_1) but also number of bedrooms (x_2), number of bathrooms (x_3), number of floors (x_4), year built (x_5), zip code (x_6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.

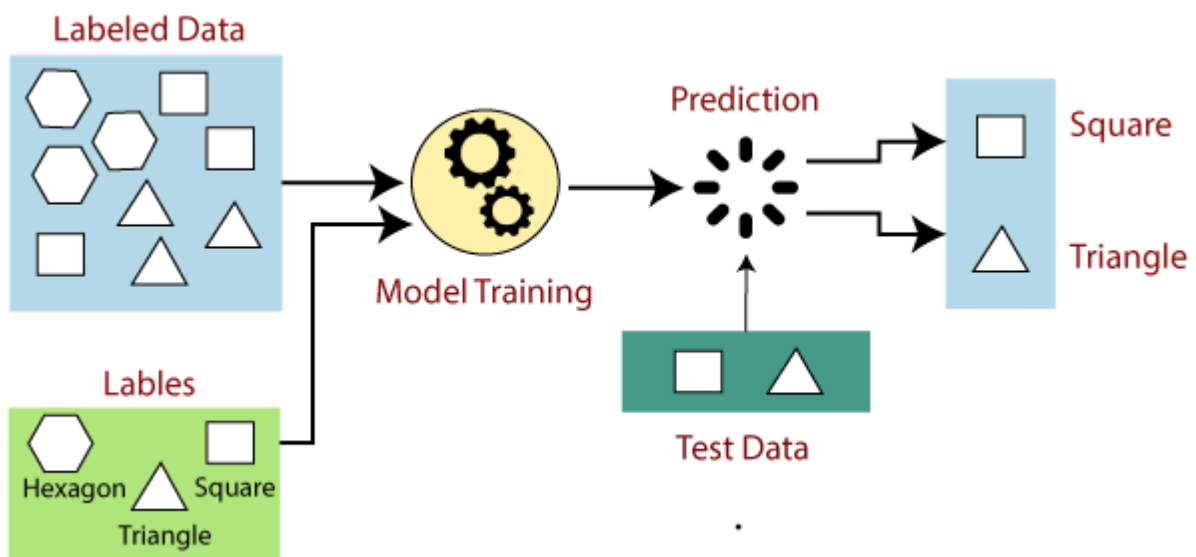


Fig 1.1 Supervised prediction model

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough

knowledge so that the model can accurately predict the output.

- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

1. Linear Regression
2. Regression Trees
3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial Regression

Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

EXISTING SYSTEM

The NEURAL ENQUIRER — a neural network architecture for answering natural language (NL) questions based on a knowledge base (KB) table. Unlike existing work

on end-to-end training of semantic parsers [Pasupat and Liang, 2015; Neelakantan et al., 2015], NEURAL ENQUIRER is fully “Neuralised”: it finds distributed representations of queries and KB tables, and executes queries through a series of neural network components called “executors”. Executors model query operations and compute intermediate execution results in the form of table annotations at different levels. NEURAL ENQUIRER can be trained with gradient descent, with which the representations of queries and the KB table are jointly optimized with the query execution logic. The training can be done in an end-to-end fashion, and it can also be carried out with stronger guidance, e.g., step-by-step supervision for complex queries. NEURAL ENQUIRER is one step towards building neural network systems that can understand natural language in real-world tasks. As a proof-of-concept, we conduct experiments on a synthetic QA task, and demonstrate that the model can learn to execute reasonably complex NL queries on small-scale KB tables.

Disadvantages

- This takes more time to process.
- The efficiency is less that compare to proposed one.

PROPOSED SYSTEM

NADAQ consists of three major components, including data storage module, model management module and user interface module. Data storage module includes MySQL as the underlying database engine, which extracts meta-data from the tables for translation model training and executes the SQL queries to return search results to human users.

Advantages

- It simplify the process.
- Minimum Processing time.

CHAPTER 2

LITERATURE SURVEY

Title : Driver Distraction Detection Using Semi-Supervised Machine Learning.

Author : Tianchi Liu ; Yan Yang ; Guang-Bin Huang ; Yong Kiang Yeo ; Zhiping Lin

Explanation

Real-time driver distraction detection is the core to many distraction countermeasures and fundamental for constructing a driver-centered driver assistance system. While data-driven methods demonstrate promising detection performance, a particular challenge is how to reduce the considerable cost for collecting labeled data. This paper explored semi-supervised methods for driver distraction detection in real driving conditions to alleviate the cost of labeling training data. Laplacian support vector machine and semi-supervised extreme learning machine were evaluated using eye and head movements to classify two driver states: attentive and cognitively distracted. With the additional unlabeled data, the semi-supervised learning methods improved the detection performance (G-mean) by 0.0245, on average, over all subjects, as compared with the traditional supervised methods. As unlabeled training data can be collected from drivers' naturalistic driving records with little extra resource, semi-supervised methods, which utilize both labeled and unlabeled data, can enhance the efficiency of model development in terms of time and cost.

Title : Detection of Driver Cognitive Distraction: A Comparison Study of Stop-Controlled Intersection and Speed-Limited Highway.

Author : Yuan Liao ; Shengbo Eben Li ; Wenjun Wang ; Ying Wang ; Guofa Li ; Bo Cheng

Explanation

Driver distraction has been identified as one major cause of unsafe driving. The existing studies on cognitive distraction detection mainly focused on high-speed driving situations, but less on low-speed traffic in urban driving. This paper presents a method for the detection of driver cognitive distraction at stop-controlled intersections and compares its feature subsets and classification accuracy with that on a speed-limited

highway. In the simulator study, 27 subjects were recruited to participate. Driver cognitive distraction is induced by the clock task that taxes visuospatial working memory. The support vector machine (SVM) recursive feature elimination algorithm is used to extract an optimal feature subset out of features constructed from driving performance and eye movement. After feature extraction, the SVM classifier is trained and cross-validated within subjects. On average, the classifier based on the fusion of driving performance and eye movement yields the best correct rate and F-measure (correctrate = $95.8 \pm 4.4\%$; for stop-controlled intersections and correct rate = $93.7 \pm 5.0\%$; for a speed-limited highway) among four types of the SVM model based on different candidate features. The comparisons of extracted optimal feature subsets and the SVM performance between two typical driving scenarios are presented.

Title : Online Driver Distraction Detection Using Long Short-Term Memory.

Author : Martin Wollmer ; Christoph Blaschke ; Thomas Schindl ; Björn Schuller ; Berthold Farber ; Stefan Mayer ; Benjamin Trefflich

Explanation

This paper presents a new computational framework for early detection of driver distractions (map viewing) using brain activity measured by electroencephalographic (EEG) signals. Compared with most studies in the literature, which are mainly focused on the classification of distracted and nondistracted periods, this study proposes a new framework to prospectively predict the start and end of a distraction period, defined by map viewing. The proposed prediction algorithm was tested on a data set of continuous EEG signals recorded from 24 subjects. During the EEG recordings, the subjects were asked to drive from an initial position to a destination using a city map in a simulated driving environment. The overall accuracy values for the prediction of the start and the end of map viewing were 81% and 70%, respectively. The experimental results demonstrated that the proposed algorithm can predict the start and end of map viewing with relatively high accuracy and can be generalized to individual subjects. The outcome of this study has a high potential to improve the design of future intelligent navigation systems. Prediction of the start of map viewing can be used to provide route information based on a driver's needs and consequently avoid map-viewing activities.

Prediction of the end of map viewing can be used to provide warnings for potential long map-viewing durations. Further development of the proposed framework and its applications in driver-distraction predictions are also discussed.

Title :Online Prediction of Driver Distraction Based on Brain Activity Patterns.

Author : Shouyi Wang ; Yiqi Zhang ; Changxu Wu ; Felix Darvas ; Wanpracha Art Chaovalitwongse

Explanation

This method proposes a method for learning joint embeddings of images and text using a two-branch neural network with multiple layers of linear projections followed by nonlinearities. The network is trained using a largemargin objective that combines cross-view ranking constraints with within-view neighborhood structure preservation constraints inspired by metric learning literature. Extensive experiments show that our approach gains significant improvements in accuracy for image-to-text and text to-image retrieval. Our method achieves new state-of-the-art results on the Flickr30K and MSCOCO image-sentence datasets and shows promise on the new task of phrase localization on the Flickr30K Entities dataset.

Title : A Gaze-Based Driver Distraction Warning System and Its Effect on Visual Behavior.

Author: Christer Ahlstrom; Katja Kircher; Albert Kircher

Explanation

Driver distraction is a contributing factor to many crashes; therefore, a real-time distraction warning system should have the potential to mitigate or circumvent many of these crashes. The objective of this paper is to investigate the usefulness of a real-time distraction detection algorithm called AttenD. The evaluation is based on data from an extended field study comprising seven drivers who drove on an average of 4351 ± 2181 km in a naturalistic setting. Visual behavior was investigated both on a global scale and on a local scale in the surroundings of each warning. An increase in the percentage of glances at the rear-view mirror and a decrease in the amount of glances at the center console were found. The results also show that visual time sharing decreased in

duration from 9.94 to 9.20 s due to the warnings, that the time from fully attentive to warning decreased from 3.20 to 3.03 s, and that the time from warning to full attentiveness decreased from 6.02 to 5.46 s. The limited number of participants does not allow any generalizable conclusions, but a trend toward improved visual behavior could be observed. This is a promising start for further improvements of the algorithm and the warning strategy.

CHAPTER 3

METHODOLOGY

The entire procedure to develop an application that meet our needs and helps us achieving our project aim can be described as pre-processing of data which includes various steps in itself and to know of the project requirements and validations and scope of the project.

SYSTEM PROCESS

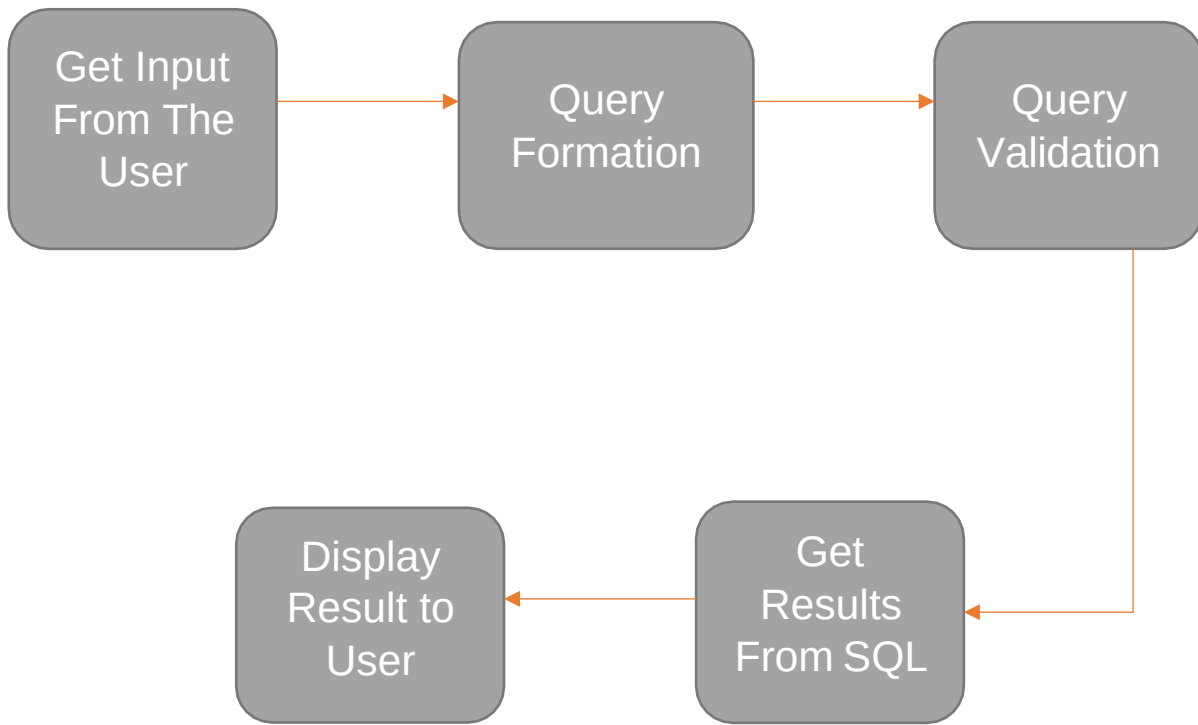


Fig 3.1 System Architecture

- **Get input from the user** : In this step the model gets the input data from the user. It asks required data from the user in the form of integers or strings or alpha numeric depending on the field attribute. The type of data to be entered in the field is already mentioned by the developer. The input data can then be used to perform various calculations and operations on them. All the necessary and mandatory fields can be mentioned and the data can then be flowed towards the process in the model.
- **Query formation** : The input data in the form of alphanumeric string is then

processed to give an sql query. The question formed is of appropriate meaning and the formed question is based on the user input string's keywords. The keywords play a major role in formation of sql query, depending on them the system defines a valid sql query that can be further processed and can derive the user requested details.

- **Query validation** : This phase takes care of the validation of the generated sql query from the previous phases. The query validation is a necessary step in the model as it eliminates any unwanted or meaningless sql query, which may raise difficulties in the system. The invalid queries thus raise the user an error message. The valid sql queries are then sent to the next step of process.
- **Get results from sql** : The valid sql queries are then processed and the requirements are then gathered. The keywords in the question describe the query to grant or access data from a requested table. The data retrieved is based on the conditions in the query. The data is retrieved from the database to which the system is connected to.
- **Display results to user** : The retrieved data from the database is then processed to display to the front end user. The data displayed is in the form of table and lists.

SYSTEM DESIGN

Loading the Live Data

Text mining is the application of natural language processing techniques and analytical methods to text data in order to derive relevant information. Google is a gold mine of data, it constitutes a rich source that can be used for capturing information about any topic imaginable. This data can be used in different use cases such as finding trends related to a specific keyword, measuring brand sentiment, and gathering feedback about new products and services.

Data Pre-Processing

Data pre-processing is an important step to prepare the data to form a Schizophrenic Discourse model. There are many important steps in data preprocessing, such as data cleaning, data transformation, and feature selection. Schizophrenic Discourse Data cleaning and transformation are methods used to remove outliers and standardize the data so that they take a form that can be easily used to create a model. A Schizophrenic Discourse data set may contain hundreds of variables (descriptors); however, many of these variables will contain redundant data. In order to simplify the dimensionality of the model, it is important to select only variables that contain unique and important information. Data mining procedures can be used to remove variables that do not contribute to the Schizophrenic Discourse model.

Tasks in data pre-processing

- **Data cleaning:** fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
- **Data integration:** using multiple databases, data cubes, or files.
- **Data transformation:** normalization and aggregation.
- **Data reduction:** reducing the volume but producing the same or similar analytical results.
- **Data discretization:** part of data reduction, replacing numerical attributes with nominal ones.

Many words in Schizophrenic Discourse dataset recur very frequently but are essentially meaningless as they are used to join words together in a sentence. It is commonly understood that stop words do not contribute to the context or content of textual documents. Due to their high frequency of occurrence, their presence in text mining presents an obstacle in understanding the content of the documents. Stop words are very frequently used common words like 'and', 'are', 'this' etc. They are not useful in classification of documents. So they must be removed. However,

the development of such stop words list is difficult and inconsistent between textual sources. This process also reduces the text data and improves the Schizophrenic Discourse system performance. Every text Schizophrenic Discourse dataset deals with these words which are not necessary for text mining applications.

: Building the Model

A LSTM network is a kind of recurrent neural network. A recurrent neural network is a neural network that attempts to model time or sequence dependent behaviour – such as language, stock prices, electricity demand and so on. This is performed by feeding back the output of a neural network layer at time t to the input of the same network layer at time $t + 1$.

An LSTM network is a recurrent neural network that has LSTM cell blocks in place of our standard neural network layers. These cells have various components called the input gate, the forget gate and the output gate.

This memory allows the network to learn long-term dependencies in a sequence which means it can take the entire context into account when making a prediction, whether that be the next word in a sentence, a sentiment classification, or the next temperature measurement. A RNN is designed to mimic the human way of processing sequences: we consider the entire sentence when forming a response instead of words by themselves.

: Training The Model

The Long Short-Term Memory network, or LSTM network, is a recurrent neural network that is trained using Backpropagation Through Time and overcomes the vanishing gradient problem. As such, it can be used to create large recurrent networks that in turn can be used to address difficult sequence problems in machine learning and achieve state-of-the-art results. Instead of neurons, LSTM networks have memory blocks that are connected through layers.

A block has components that make it smarter than a classical neuron and a memory for recent sequences. A block contains gates that manage the block's state and output. A block operates upon an input sequence and each gate within a block uses the sigmoid activation units to control whether they are triggered or not, making the change of state and addition of information flowing through the block conditional.

There are three types of gates within a unit:

- **Forget Gate:** conditionally decides what information to throw away from the block.
- **Input Gate:** conditionally decides which values from the input to update the memory state.
- **Output Gate:** conditionally decides what to output based on input and the memory of the block. Each unit is like a mini-state machine where the gates of the units have weights that are learned during the training procedure.

LSTMs can be used to model univariate time series forecasting problems. These are problems comprised of a single series of observations and a model is required to learn from the series of past observations to predict the next value in the sequence. The LSTM model will learn a function that maps a sequence of past observations as input to an output observation. As such, the sequence of observations must be transformed into multiple examples from which the LSTM can learn.

The Long Short Term Memory neural network is a type of a Recurrent Neural Network (RNN). RNNs use previous time events to inform the later ones. For example, to classify what kind of event is happening in a movie, the model needs to use information about previous events. RNNs work well if the problem requires only recent information to perform the present task. If the problem requires long term dependencies, RNN would struggle to model it. The LSTM was designed to learn long term dependencies. It remembers the information for long periods.

SOFTWARE DEVELOPMENT

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

Waterfall Model

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

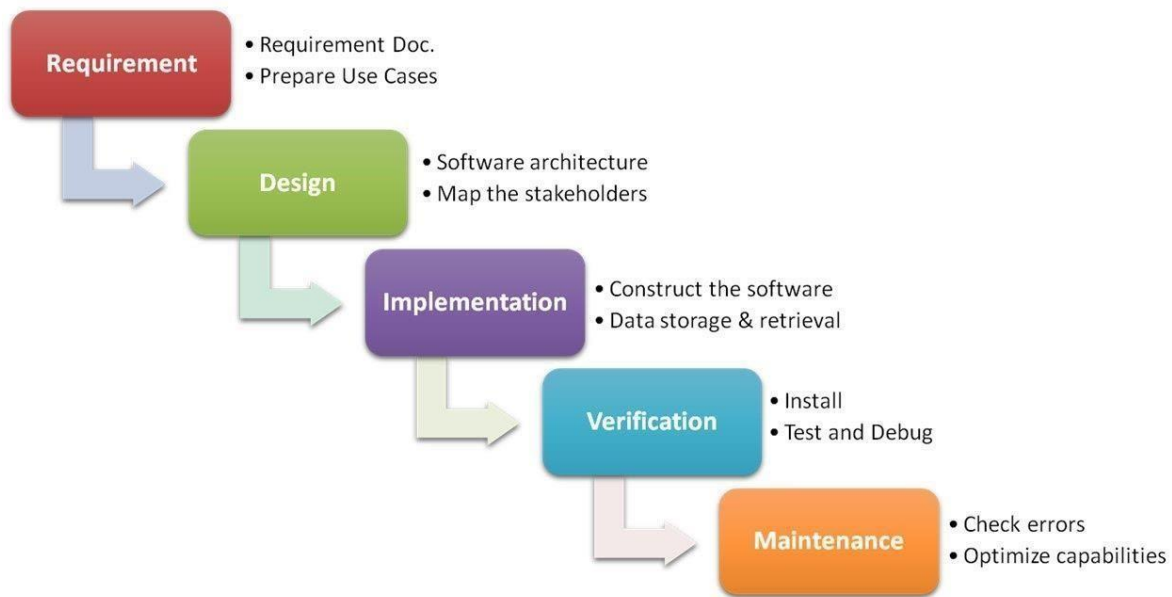


Fig 3.2 Waterfall Project Management Methodology

- **Requirement gathering and documentation** : In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.
- **System design** : Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

- **Implementation** : During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.
- **Testing** : Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for re-evaluation.
- **Delivery/Deployment** : In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.
- **Maintenance** : The final solution has been implemented to the client and is being used. As troubles arise, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

RAD Model

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.

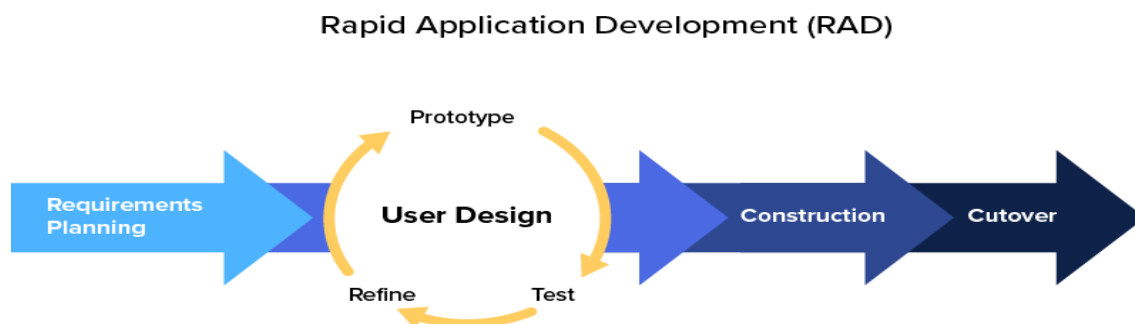


Fig 3.3 Rapid Application Development Methodology

- **Phase 1: Requirements planning** : This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project. During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

- Researching the current problem.
- Defining the requirements for the project.
- Finalizing the requirements with each stakeholder's approval.

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

- **Phase 2 : User design** : Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations. This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations. All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design. Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

- **Phase 3: Rapid construction** : This phase takes the prototypes and beta systems from the design phase and converts them into the working model. Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

- Preparation for rapid construction.
- Program and application development.
- Coding.
- Unit, integration, and system testing.

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives. This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

- **Phase 4: Cutover** : This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training. All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD Methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Advantages of RAD Methodology

RAD lets you break the project down into smaller, more manageable tasks. The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience. Clients get a working product delivered in a shorter time frame. Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

LANGUAGE DESCRIPTION

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

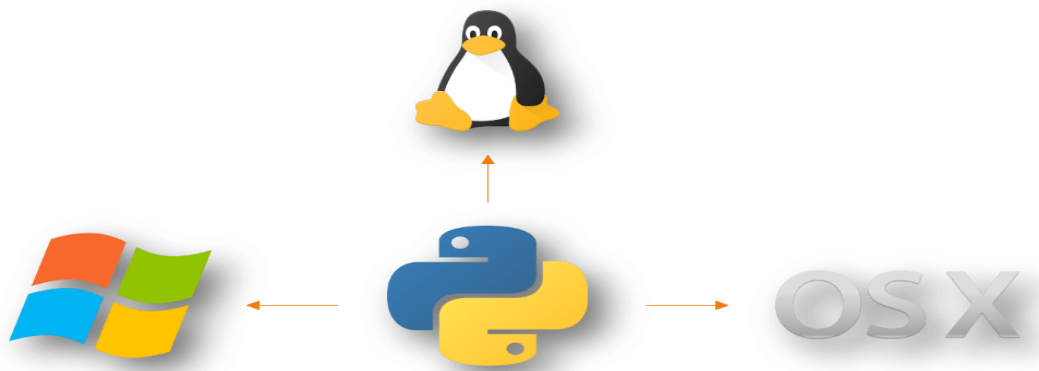


Fig 3.4 Python Supporting Operating Systems

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations. In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries
- Simple & support IoT
- And many more

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

- **Explicit Line Joining** : In explicit line joining, we use a backward slash to split a statement into a multiline statement.
- **Implicit Line Joining** : Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.
- **Multiple Statements on a Single Line** : In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.
- **Whitespaces and Indentation** : Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python. The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code. Following are different kinds of comments that can be included in our Python program:

- **Single Line Comments** : Single line Python comments are marked with # character. These comments end at the end of the physical line, which means that all characters starting after the # character (and lasts till the end of the line) are part of the comment.

- **Docstring Comments** : Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.
- **Multiline Comments** : It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.
- **Python Block Comments** : We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

- **About Pandas** : Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make

data manipulation and analysis easy, among many other things. The DataFrame is one of these structures. Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn. Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily. Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' Data Frames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and Data Frames.

1. **Series** : A pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.
2. **DataFrame** : A pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc. Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of

potentially different types. In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is: a Pandas DataFrame

- A Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- A NumPy ndarray, which can be a record or structured
- A two-dimensional ndarray
- Dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows: It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.

- Merging and joining data sets.
 - Reshaping and pivoting data sets.
 - Aligning data and dealing with missing data.
 - Manipulating data using integrated indexing for DataFrame objects.
 - Performing split-apply-combine on data sets using the group by engine.
 - Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
 - Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
 - Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
 - Reading from files with CSV, XLSX, TXT, among other formats.
 - Arranging data in an order ascending or descending.
 - Filtering data around a condition.
 - Analyzing time series.
 - Iterating over a data set.
- **Numpy** : Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy. A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension. NumPy is, just like SciPy,

Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient. NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays. an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in

data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

- **Matplotlib** : Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib. Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams. Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well. Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib. Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

Selection of Integrated Development Environment

IDE - Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications. Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer. Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- ✓ NetBeans
- ✓ PyCharm
- ✓ Liclipse (Eclipse)

LICLIPSE - Liclipse is at the core for all Python based application development activities. It's been available for many years. Liclipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of Liclipse

- Perfect Debugger

- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature
- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

Selection of Operating System

There are various open source and commercial Operating systems are available in the market. Computer Operating systems can be broadly classified into two types:

- Normal Operating System
- Real Time Operating System

Windows 10 is a is the most powerful Microsoft operating system it can be used for desktop/laptop computers as well as IoT devices.

Important Features That Make Windows 10 our Operating System of Choice

- The loading time is very fast compare to other OS
- Supports built-in mobile device executive facilities
- Stability and performance
- Have hologram technology

Selection of Web Application Framework

Framework provides a structure for application development. This makes web application programmers' lives easier when developing consistent, accessible, and workable enterprise wide web applications. They automate the implementation of redundant tasks or extensions for common operations, reduces the development and testing time and allowing programmers to concentration more on application logic instead of routine works.

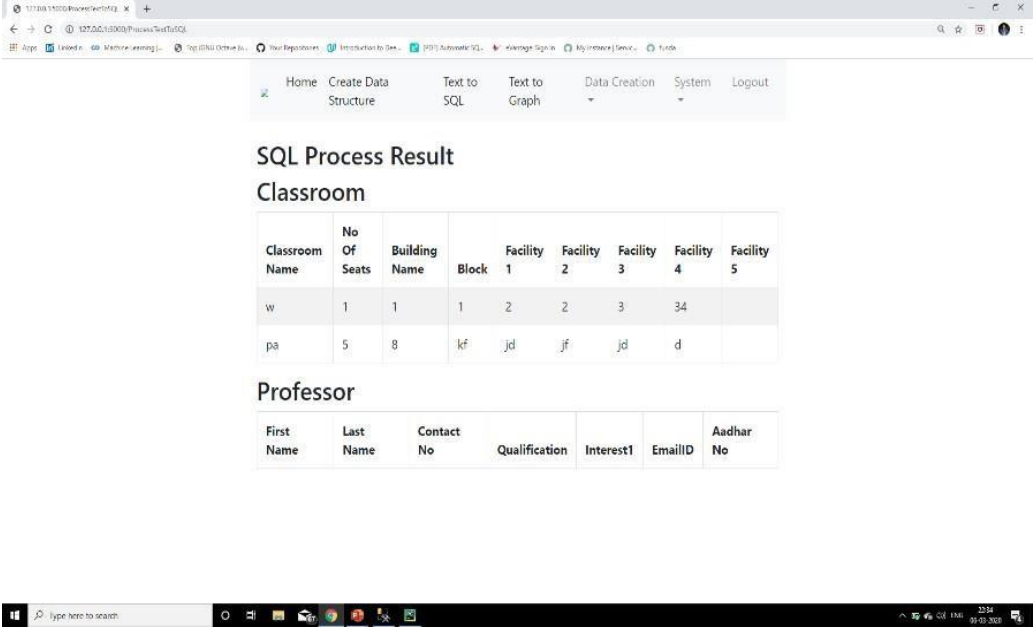
Python framework can be classified as two types:

- **Full-Stack Framework** : The full-stack systems give total help to engineers, including vital components, for example, structure approval, structure generators, and format designs.
- **Non-Full Stack** : The non-full-stack systems don't give extra functionalities and highlights to the clients. Designers need to include a great deal of code and different things physically.
 - **Microframework** : Miniaturized scale structures are little, straightforward, and simple to utilize. They are laconic and have straightforward documentation. URL directing is frequently RESTful web services. Miniaturized scale systems use WSGI and work through HTTP request/Response. They are a decent decision for little applications, or as a feature of a bigger undertaking.
 - **Asynchronous Framework** : In situations where solicitation handling pace assumes a significant job or an undertaking should manage long reaction times, nonconcurrent structure ought to work.

CHAPTER 4

RESULTS AND DISCUSSION

The system that has been generated will be a greater help to users with no knowledge in how SQL works or how to retrieve data from a database. The system provides a friendly user interface to the user who tries to access. It requests user with the basic knowledge on English grammar to input his question based on his requirements. The system then process that input and checks with the validity and forms the SQL query. The query is then validated and checked for its correctness. If the query is invalid the query gets rejected and the user has to input a new string of valid requirements. If the query gets accepted then the system process the query tables and rows based on the given requirement, the system then access the database and retrieves the data that needs to be accessed. The data that has been accessed and retrieved is then processed to display to the end user. The display of output is the valid records and attributes from the database. The results are then displayed to the end user in the user interface.



The screenshot shows a web browser window with a URL of 127.0.0.1:5000/ProcessTextToSQL. The application has a navigation bar with links: Home, Create Data Structure, Text to SQL, Text to Graph, Data Creation, System, and Logout. Below the navigation bar, the page title is 'SQL Process Result'. There are two sections: 'Classroom' and 'Professor'. The 'Classroom' section displays a table with columns: Classroom Name, No Of Seats, Building Name, Block, Facility 1, Facility 2, Facility 3, Facility 4, and Facility 5. The 'Professor' section displays a table with columns: First Name, Last Name, Contact No, Qualification, Interest1, EmailID, and Aadhar No.

Classroom Name	No Of Seats	Building Name	Block	Facility 1	Facility 2	Facility 3	Facility 4	Facility 5
w	1	1	1	2	2	3	34	
pa	5	8	kf	jd	jf	jd	d	

First Name	Last Name	Contact No	Qualification	Interest1	EmailID	Aadhar No

Fig 4.1 Classroom table result

In the general workflow of the system, users are allowed to ask queries in text. Once the user enters the text input, NADAQ executes the query and present the results in the next page of the interface. If NADAQ decides to reject the query due to the limited

relevance of the original question to the contents of the database, it displays no results in the user interface.

SQL Process Result

Classroom

Classroom Name	No Of Seats	Building Name	Block	Facility 1	Facility 2	Facility 3	Facility 4	Facility 5

Professor

First Name	Last Name	Contact No	Qualification	Interest1	EmailID	Aadhar No
pava	jnfd	iuhd	jnfd	jbf	uhd	uih@123.com
ani	jbd	kjbd	jbd	ubd	ubc	kjba@gmail.com

Fig 4.2 Professor table result

Although several methodologies are employed to extract information from a database, Natural Language Processing has set a new standard in doing the same. This work presents a clear picture on the steps that are involved in NLP. Various processes like tokenization, lemmatization, syntactic and semantic analysis are carried out to generate an equivalent SQL query from a natural language query. Following are the future improvements that can be incorporated; The input received can be of audio form, which can be converted into textual format; The SQL query could be of greater complexity; The database could be larger in terms of attributes and tuples. Also, there could be multiple tables of related data which can be accessed using JOIN keyword; It could be used to create chatbots for various sectors which handle large databases and can help users to access them with greater ease; The output could be converted as a sentence then into audio format to make the system more interactive; This work can also be extended to other languages. To conclude, NLP is a boon to any ordinary person having no knowledge on database management. In short, NLP proves to be a promising tool for future advancements in numerous fields like medicine, business, and education so on.

Given a natural language query, our system first translates it to a SQL statement and then evaluates it against an RDBMS. To achieve high reliability, our system explains to the user how her query is actually processed. Using our system, even naive users are able to accomplish logically complex query tasks, in which the target SQL statements include comparison predicates, conjunctions, quantifications, multi-level aggregations, among other things.

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION

Here we demonstrate our new natural language database querying (NADAQ) system. The design of NADAQ combines the state-of-the-art deep learning techniques and the mature SQL parsing techniques. It enables the system to significantly improve the quality of query translation and inspire new solutions to meaningless question rejection and query candidate recommendation. The results show NADAQ provides nearly commercial standard service as a querying interface to human users, even when they neither understand SQL nor comprehend the database schema.

FUTURE WORK

The future improvements that can be incorporated; The input received can be of audio form, which can be converted into textual format; The SQL query could be of greater complexity; The database could be larger in terms of attributes and tuples. Also, there could be multiple tables of related data which can be accessed using JOIN keyword; It could be used to create chatbots for various sectors which handle large databases and can help users to access them with greater ease; The output could be converted as a sentence then into audio format to make the system more interactive; This work can also be extended to other languages. To conclude, NLP is a boon to any ordinary person having no knowledge on database management. In short, NLP proves to be a promising tool for future advancements in numerous fields like medicine, business, and education so on.

We can add more synonyms for column names and table names so that system is able to handle more queries. To provides some recommendations so that it is helpful for user. In future we can add some strong recommendation framework in this system so that user will have to take fewer efforts. The system also stores the successfully executed queries. It can use static database, if we want to add any other table in database we also have to add grammar to handle queries for that table as grammar is hard coded but we can also remove this problem by constructing a dynamic framework in which user can dynamically add new tables and remove older ones. In this

architecture we have to generate grammar dynamically which can be future enhancement for this system.

A future idea to convert a Hindi speech into SQL query. This is first done by converting the sentence into English text, followed by formation of SQL Query, with the help of several python libraries like Goslate, NLTK. After query formation, the data is extracted from the database. The project objective is to provide accessibility to handle data without having any specific background in Computer Science. The further advancement and enhancement of the functions and keywords will be seen in future.

The system can be further developed by incorporating the following future works, which are yet to be implemented in the present system. The development on the points mentioned in future work is in progress.

- The system can be developed to handle qualitative quantifiers in the user query.
- A recursive algorithm can be designed to handle advanced nested queries which have an independent SQL query in the condition clause.
- Machine Learning algorithms can be incorporated to determine the most efficient query amongst all the possible SQL queries for a user query.
- There can be a situation where the user query has more than one semantic. An algorithm can be designed to formulate a SQL query with respect to each derived semantic.
- The system uses only MySQL database. It can be broadly developed to accept other database or unstructured database systems.

REFERENCES

- [1] K. Cho et al., “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in Proc. EMNLP, 2014, pp. 1724–1734.
- [2] L. Dong and M. Lapata, “Language to logical form with neural attention,” in Proc. ACL, 2016, pp. 33–43.
- [3] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in Proc. ICML, 2017, pp. 1243–1252.
- [4] K. Guu, P. Pasupat, E. Z. Liu, and P. Liang, “From language to programs: Bridging reinforcement learning and maximum marginal likelihood,” in Proc. ACL, 2017, pp. 1051–1062.
- [5] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.
- [6] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, “Learning a neural semantic parser from user feedback,” in Proc. ACL, 2017, pp. 963–973.
- [7] R. Jia and P. Liang, “Data recombination for neural semantic parsing,” in Proc. ACL, 2016, pp. 12–22.
- [8] A. Kannan et al., “Smart reply: Automated response suggestion for Email,” in Proc. KDD, 2016, pp. 955–964.
- [9] J. Lafferty et al., “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in Proc. 18th Int. Conf. Mach. Learn. (ICML), vol. 1, 2001, pp. 282–289.
- [10] F. Li and H. V. Jagadish, “Constructing an interactive natural language interface for relational databases,” Proc. VLDB Endowment, vol. 8, no. 1, pp. 73–84, 2014.
- [11] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, “Neural symbolic machines: Learning semantic parsers on freebase with weak supervision,” in Proc. ACL, 2017, pp. 23–33.

APPENDIX

A.SOURCE CODE

```
import uuid
from io import StringIO

from flask import Flask, request, render_template, redirect, url_for, session
import os
import pypyodbc
import zipfile
import numpy as np
import cv2
from os.path import isfile, join

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')
import seaborn as sns
sns.set()

from RoleModel import RoleModel
from UserModel import UserModel
from Constants import connString

app = Flask(_name_)
app.secret_key = "MySecret"
ctx = app.app_context()
ctx.push()
with ctx:
    pass
userName = ""
roleObject = None
message = ""
msgType = ""

def initialize():
    global message, msgType
    message = ""
    msgType=""

def processRole(optionID):
    if optionID == 10 :
        if roleObject.canRole == False :
            return False
    if optionID == 20 :
```

```

        if roleObject.canUser == False :
            return False
    if optionID == 30 :
        if roleObject.CL111 == False :
            return False
    if optionID == 40 :
        if roleObject.CL222 == False :
            return False
    if optionID == 50 :
        if roleObject.CL333 == False :
            return False
    return True

@app.route('/')
def index():
    global userID, userName
    return render_template('Login.html') # when the home page is called Index.html
will be triggered.

@app.route('/processLogin', methods=['POST'])
def processLogin():
    global userID, userName, roleObject
    userName= request.form['userName']
    password= request.form['password']
    conn1 = pypyodbc.connect(connString, autocommit=True)
    cur1 = conn1.cursor()
    sqlcmd1 = "SELECT * FROM UserTable WHERE userName = '"+userName+"'
AND password = '"+password+"' AND isActive = 1";

    cur1.execute(sqlcmd1)
    row = cur1.fetchone()

    cur1.commit()
    if not row:
        return render_template('Login.html', processResult="Invalid Credentials")
    userID = row[0]
    userName = row[3]

    cur2 = conn1.cursor()
    sqlcmd2 = "SELECT * FROM Role WHERE RoleID = '"+str(row[6])+"'";
    cur2.execute(sqlcmd2)
    row2 = cur2.fetchone()

    if not row2:
        return render_template('Login.html', processResult="Invalid Role")

    roleObject = RoleModel(row2[0], row2[1],row2[2],row2[3],row2[4],row2[5]

```

```

    return render_template('Dashboard.html')

@app.route("/ChangePassword")
def changePassword():
    global userID, userName
    return render_template('ChangePassword.html')

@app.route("/ProcessChangePassword", methods=['POST'])
def processChangePassword():
    global userID, userName
    oldPassword= request.form['oldPassword']
    newPassword= request.form['newPassword']
    confirmPassword= request.form['confirmPassword']
    conn1 = pypyodbc.connect(connString, autocommit=True)
    cur1 = conn1.cursor()
    sqlcmd1 = "SELECT * FROM UserTable WHERE userName = '"+userName+"'
AND password = '"+oldPassword+"'";
    cur1.execute(sqlcmd1)
    row = cur1.fetchone()
    cur1.commit()
    if not row:
        return render_template('ChangePassword.html', msg="Invalid Old Password")

    if newPassword.strip() != confirmPassword.strip() :
        return render_template('ChangePassword.html', msg="New Password and
Confirm Password are NOT same")

    conn2 = pypyodbc.connect(connString, autocommit=True)
    cur2 = conn2.cursor()
    sqlcmd2 = "UPDATE UserTable SET password = '"+newPassword+"' WHERE
userName = '"+userName+"'";
    cur1.execute(sqlcmd2)
    cur2.commit()
    return render_template('ChangePassword.html', msg="Password Changed
Successfully")

@app.route("/Dashboard")
def Dashboard():
    global userID, userName
    return render_template('Dashboard.html')

@app.route("/Information")
def Information():
    global message, msgType
    return render_template('Information.html', msgType=msgType, message =
message)
@app.route("/UserListing")

```

```

def UserListing():
    global userID, userName

    global message, msgType, roleObject
    if roleObject == None:
        message = "Application Error Occurred"
        msgType="Error"
        return redirect(url_for('Information'))
    canRole = processRole(10)

    if canRole == False:
        message = "You Don't Have Permission to Access User"
        msgType="Error"
        return redirect(url_for('Information'))

    conn2 = pypyodbc.connect(connString, autocommit=True)
    cursor = conn2.cursor()
    sqlcmd1 = "SELECT * FROM UserTable ORDER BY userName"
    cursor.execute(sqlcmd1)
    records = []

    while True:
        dbrow = cursor.fetchone()
        if not dbrow:
            break

        conn3 = pypyodbc.connect(connString, autocommit=True)
        cursor3 = conn3.cursor()
        temp = str(dbrow[6])
        sqlcmd3 = "SELECT * FROM Role WHERE RoleID = '"+temp+"'"
        cursor3.execute(sqlcmd3)
        rolerow = cursor3.fetchone()
        roleModel = RoleModel(0)
        if rolerow:
            roleModel = RoleModel(rolerow[0],rolerow[1])
        else:
            print("Role Row is Not Available")

        row = UserModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[4], dbrow[5],
            dbrow[6], roleModel=roleModel)
        records.append(row)
    return render_template('UserListing.html', records=records)

@app.route("/UserOperation")
def UserOperation():
    global userID, userName

```



```

        rolerow = cursor3.fetchone()
        roleModel = RoleModel(0)
        if rolerow:
            roleModel = RoleModel(rolerow[0],rolerow[1])
        else:
            print("Role Row is Not Available")
        row = UserModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[4], dbrow[5],
dbrow[6], roleModel=roleModel)

```

```

    return render_template('UserOperation.html', row = row, operation=operation,
rolesDDLlist=rolesDDLlist )

```

```

@app.route("/ProcessUserOperation",methods = ['POST'])

```

```

def processUserOperation():

```

```

    global userName, userID

```

```

    operation = request.form['operation']

```

```

    unqid = request.form['unqid'].strip()

```

```

    userName= request.form['userName']

```

```

    emailid= request.form['emailid']

```

```

    password=request.form['password']

```

```

    contactNo= request.form['contactNo']

```

```

    isActive = 0

```

```

    if request.form.get("isActive") != None :

```

```

        isActive = 1

```

```

    roleID= request.form['roleID']

```

```

    conn1 = pypyodbc.connect(connString, autoccommit=True)

```

```

    cur1 = conn1.cursor()

```

```

    if operation == "Create" :

```

```

        sqlcmd = "INSERT INTO UserTable( userName,emailid, password,contactNo,
isActive, roleID) VALUES('"+userName+"','"+emailid+"', '"+contactNo+"',
 '"+password+"', '"+str(isActive)+"', '"+str(roleID)+"')

```

```

    if operation == "Edit" :

```

```

        sqlcmd = "UPDATE UserTable SET userName = '"+userName+"', emailid =
 '"+emailid+"', password = '"+password+"',contactNo='"+contactNo+"', isActive =
 '"+str(isActive)+"', roleID = '"+str(roleID)+"' WHERE UserID = '"+unqid+"'"

```

```

    if operation == "Delete" :

```

```

        sqlcmd = "DELETE FROM UserTable WHERE UserID = '"+unqid+"'"

```

```

    if sqlcmd == "" :

```

```

        return redirect(url_for('Information'))

```

```

    cur1.execute(sqlcmd)

```

```

    cur1.commit()

```

```

    conn1.close()

```

```

    return redirect(url_for("UserListing"))

```

```

'''

```



```

if roleObject == None:
    message = "Application Error Occurred"
    msgType="Error"
    return redirect(url_for('/'))
canRole = processRole(120)

if canRole == False:
    message = "You Don't Have Permission to Access Role"
    msgType="Error"
    return redirect(url_for('Information'))

operation = request.args.get('operation')
unqid = ""
row = RoleModel(0, "",0,0,0,0)
if operation != "Create" :
    unqid = request.args.get('unqid').strip()

    conn2 = pypyodbc.connect(connString, autocommit=True)
    cursor = conn2.cursor()
    sqlcmd1 = "SELECT * FROM Role WHERE RoleID = '"+unqid+"'"
    cursor.execute(sqlcmd1)
    while True:
        dbrow = cursor.fetchone()
        if not dbrow:
            break
        row =
RoleModel(dbrow[0],dbrow[1],dbrow[2],dbrow[3],dbrow[4],dbrow[5],dbrow[6])

    return render_template('RoleOperation.html', row = row, operation=operation )

@app.route("/ProcessRoleOperation", methods=['POST'])
def ProcessRoleOperation():
    global message, msgType
    if roleObject == None:
        message = "Application Error Occurred"
        msgType="Error"
        return redirect(url_for('/'))
    canRole = processRole(120)

    if canRole == False:
        message = "You Don't Have Permission to Access Role"
        msgType="Error"
        return redirect(url_for('Information'))
    print("ProcessRole")

    operation = request.form['operation']

```

```

if operation != "Delete" :
    roleName = request.form['roleName']
    canRole = 0
    canUser = 0
    CL111 = 0
    CL222 = 0
    CL333 = 0
    if request.form.get("canRole") != None :
        canRole = 1
    if request.form.get("canUser") != None :
        canUser = 1
    if request.form.get("CL111") != None :
        CL111 = 1
    if request.form.get("CL222") != None :
        CL222 = 1
    if request.form.get("CL333") != None :
        CL333 = 1
print(1)
unqid = request.form['unqid'].strip()
print(operation)
conn3 = pypyodbc.connect(connString, autocommit=True)
cur3 = conn3.cursor()
sqlcmd = ""
if operation == "Create" :
    sqlcmd = "INSERT INTO Role (roleName, canRole, canUser, CL111, CL222,
CL333) VALUES ('"+roleName+"', '"+str(canRole)+"', '"+str(canUser)+"',
 '"+str(CL111)+"', '"+str(CL222)+"', '"+str(CL333)+"')
    if operation == "Edit" :
        print("edit inside")
        sqlcmd = "UPDATE Role SET roleName = '"+roleName+"', canRole =
 '"+str(canRole)+"', canUser = '"+str(canUser)+"', CL111 = '"+str(CL111)+"', CL222 =
 '"+str(CL222)+"', CL333 = '"+str(CL333)+"' WHERE RoleID = '"+unqid+"'"
    if operation == "Delete" :
        conn4 = pypyodbc.connect(connString, autocommit=True)
        cur4 = conn4.cursor()
        sqlcmd4 = "SELECT roleID FROM UserTable WHERE roleID = '"+unqid+"'"
        cur4.execute(sqlcmd4)
        dbrow4 = cur4.fetchone()
        if dbrow4:
            message = "You can't Delete this Role Since it Available in Users Table"
            msgType="Error"
            return redirect(url_for('Information'))

    sqlcmd = "DELETE FROM Role WHERE RoleID = '"+unqid+"'"
print(operation, sqlcmd)
if sqlcmd == "" :
    return redirect(url_for('Information'))

```

```

cur3.execute(sqlcmd)
cur3.commit()

return redirect(url_for('RoleListing'))

'''
    Role Operation End
'''

from database import Database
@app.route('/CreateDataStructure')
def CreateDataStructure():
    global userID, userName
    database = Database()
    database.load("abc")
    return render_template('CreateDataStructure.html')

from ClassroomModel import ClassroomModel

@app.route("/ClassroomListing")
def ClassroomListing():
    global msgText, msgType
    print("ClassroomListing Called")
    searchData = request.args.get('searchData')
    if searchData == None:
        searchData = "";
    conn2 = pypyodbc.connect(
        connString,
        autocommit=True)
    print(connString)
    cursor = conn2.cursor()
    sqlcmd1 = "SELECT * FROM Classroom WHERE ClassroomName like '" +
searchData + "%'"
    print(sqlcmd1)
    cursor.execute(sqlcmd1)
    records = []

    while True:
        dbrow = cursor.fetchone()
        if not dbrow:
            break
        row = ClassroomModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[3],
dbrow[4], dbrow[5], dbrow[6], dbrow[7])
        records.append(row)
    return render_template('ClassroomListing.html', records=records,
searchData=searchData)

```

```

@app.route("/ClassroomOperation")
def ClassroomOperation():
    global msgText, msgType
    operation = request.args.get('operation')
    unqid = ""
    row = ClassroomModel(0, "", "", "")
    row = None
    if operation != "Create":
        unqid = request.args.get('unqid').strip()
        conn2 = pypyodbc.connect(
            connString,
            autocommit=True)
        cursor = conn2.cursor()
        sqlcmd1 = "SELECT * FROM Classroom WHERE ClassroomID = " + unqid + ""
        cursor.execute(sqlcmd1)
        while True:
            dbrow = cursor.fetchone()
            if not dbrow:
                break
            row = ClassroomModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[4],
            dbrow[5], dbrow[6], dbrow[7], dbrow[8], dbrow[9])

    return render_template('ClassroomOperation.html', row=row, operation=operation)

@app.route("/ProcessClassroomOperation", methods=['POST'])
def ProcessClassroomOperation():
    global msgText, msgType
    operation = request.form['operation']
    if operation != "Delete":
        classroomName = request.form['classroomName']
        noOfSeats = request.form['noOfSeats']
        noOfComputers = request.form["noOfComputers"]
        buildingName = request.form['buildingName']
        block = request.form['block']
        facility1 = request.form['facility1']
        facility2 = request.form['facility2']
        facility3 = request.form['facility3']
        facility4 = request.form['facility4']
        facility5 = request.form['facility5']

    unqid = request.form['unqid'].strip()

    conn1 = pypyodbc.connect(
        connString,
        autocommit=True)
    cur1 = conn1.cursor()

```

```

    if operation == "Create":
        sqlcmd = "INSERT INTO Classroom (classroomName, noOfSeats,
noOfComputers, buildingName, block, facility1, facility2, facility3, facility4, facility5) "
        \
            "VALUES('" + classroomName + "', '" + str(noOfSeats) + "',
'" + str(noOfComputers) + "', '" + buildingName + "', '" + block + "', '" + facility1 + "', '" +
facility2 + "', '" +
            facility3 + "', '" + facility4 + "', '" + facility5 + "')"
    if operation == "Edit":
        print("edit inside")
        sqlcmd = "UPDATE Classroom SET classroomName = '" + classroomName + "',
noOfSeats = '" + str(noOfSeats) + "', noOfComputers = '" + str(noOfComputers) + "',
buildingName = '" + buildingName + "', facility3 = '" + facility3 + "', block = '" + block +
"', facility1 = '" + facility1 + "', facility2 = '" + facility2 + "', facility4 = '" + facility4 + "',
facility5 = '" + facility5 + "' " \
            "WHERE ClassroomID = '" + unqid + "'"
    if operation == "Delete":
        sqlcmd = "DELETE FROM Classroom WHERE ClassroomID = '" + unqid + "'"
        print(operation, sqlcmd)
        if sqlcmd == "":
            return redirect(url_for('Error'))
        cur1.execute(sqlcmd)
        cur1.commit()
        conn1.close()
        # return render_template('ClassroomListing.html', processResult="Success!!!. Data
        Uploaded. ")
        return redirect(url_for("ClassroomListing"))
@app.route("/ProfessorListing")
def ProfessorListing():
    global msgText, msgType
    print("ProfessorListing Called")
    searchData = request.args.get('searchData')
    if searchData == None:
        searchData = ""
    conn2 = pypyodbc.connect(
        connString,
        autocommit=True)
    print(connString)
    cursor = conn2.cursor()
    sqlcmd1 = "SELECT * FROM Professor WHERE firstName like '" + searchData +
    "%'"
    print(sqlcmd1)
    cursor.execute(sqlcmd1)
    records = []

    while True:
        dbrow = cursor.fetchone()

```

```

        if not dbrow:
            break
        row = ProfessorModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[3],
dbrow[4], dbrow[5], dbrow[6], dbrow[7])
        records.append(row)
    return render_template('ProfessorListing.html', records=records,
searchData=searchData)

from ProfessorModel import ProfessorModel
@app.route("/ProfessorOperation")
def ProfessorOperation():
    global msgText, msgType
    operation = request.args.get('operation')
    unqid = ""
    row = ProfessorModel(0, "", "", "")
    row = None
    if operation != "Create":
        unqid = request.args.get('unqid').strip()
        conn2 = pypyodbc.connect(
            connString,
            autocommit=True)
        cursor = conn2.cursor()
        sqlcmd1 = "SELECT * FROM Professor WHERE ProfessorID = " + unqid + ""
        cursor.execute(sqlcmd1)
        while True:
            dbrow = cursor.fetchone()
            if not dbrow:
                break
            row = ProfessorModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[4],
dbrow[5], dbrow[6], dbrow[7], dbrow[8])

    return render_template('ProfessorOperation.html', row=row, operation=operation)

@app.route("/ProcessProfessorOperation", methods=['POST'])
def ProcessProfessorOperation():
    global msgText, msgType
    operation = request.form['operation']
    if operation != "Delete":

        title = request.form['title']
        firstName = request.form['firstName']
        lastName = request.form['lastName']
        qualification = request.form['qualification']
        interest1 = request.form['interest1']
        contactNbr = request.form['contactNbr']
        emailID = request.form['emailID']
        aadharNbr = request.form['aadharNbr']

```

```

noOfYearsExperience = request.form['noOfYearsExperience']

unqid = request.form['unqid'].strip()

conn1 = pypyodbc.connect(
    connString,
    autocommit=True)
cur1 = conn1.cursor()

if operation == "Create":
    sqlcmd = "INSERT INTO Professor (title, firstName, lastName, qualification,
    interest1, contactNbr, emailID, aadharNbr, noOfYearsExperience) " \
        "VALUES(" + title + ", " + firstName + ", " + lastName + ", " + qualification
    + ", " + interest1 + ", " \
        "" + contactNbr + ", " + emailID + ", " + aadharNbr + ",
    ""+str(noOfYearsExperience)+"")"
    if operation == "Edit":
        print("edit inside")
        sqlcmd = "UPDATE Professor SET title = " + title + ", firstName = " + firstName
    + ", contactNbr = " + contactNbr + ", lastName = " + lastName + ", qualification = "
    + qualification + ", interest1 = " + interest1 + ", emailID = " + emailID + ", aadharNbr
    = " + aadharNbr + ", noOfYearsExperience = "+str(noOfYearsExperience)+" " \
        "WHERE ProfessorID = " + unqid + ""
    if operation == "Delete":
        sqlcmd = "DELETE FROM Professor WHERE ProfessorID = " + unqid + ""
    print(operation, sqlcmd)
    if sqlcmd == "":
        return redirect(url_for('Error'))
    cur1.execute(sqlcmd)
    cur1.commit()
    conn1.close()
    # return render_template('ProfessorListing.html', processResult="Success!!!. Data
    Uploaded. ")
    return redirect(url_for("ProfessorListing"))
@app.route('/TextToSQL')
def TextToSQL():
    global userID, userName
    return render_template('TextToSQL.html')

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

@app.route("/ProcessTextToSQL", methods = ['POST'])
def ProcessTextToSQL():
    global userName, userID
    etext = request.form['etext']
    etext = etext.lower()

```

```

stop_words = set(stopwords.words('english'))
stop_words.discard("all")
print(type(stop_words), stop_words)
word_tokens = word_tokenize(etext)

filtered_sentence = [w for w in word_tokens if not w in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
csql = ""
psql = ""
tsql = ""
print(etext)
where = ""
if "all" in etext and "classroom" in etext:
    csql = "SELECT * FROM classroom"
if "equal" in etext and "classroom" in etext:
    res = etext.split("equal")
    if len(res) > 0:
        where = " WHERE "
        r = res[1].strip()
        where = where + " classroomName like '" + str(r) + "%' OR "
        ridx = where.rindex("OR")
        where = where[0:ridx]
        csql = "SELECT * FROM classroom " + where
    else:
        csql = "SELECT * FROM classroom"
    print(where)
if "all" in etext and "professor" in etext:
    psql = "SELECT * FROM professor"
if "equal" in etext and "professor" in etext:
    res = etext.split("equal")
    print(len(res), "MMMMMMMMMMMMMMMMMMMMMMMMMMMMmm")
    if len(res) > 0:
        where = " WHERE "
        r = res[1].strip()
        where = where + " firstName like '" + str(r) + "%' OR "
        ridx = where.rindex("OR")
        where = where[0:ridx]
        psql = "SELECT * FROM Professor " + where
    print(where)

```

```

print(csql)
crecords = []
if csql != "":
    conn2 = pypyodbc.connect(
        connString,
        autocommit=True)
    print(connString)
    cursor = conn2.cursor()
    sqlcmd1 = csql
    print(sqlcmd1)
    cursor.execute(sqlcmd1)
    while True:
        dbrow = cursor.fetchone()
        if not dbrow:
            break
        row = ClassroomModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[3],
dbrow[4], dbrow[5], dbrow[6],
                        dbrow[7], dbrow[8])
        crecords.append(row)
print(psql)
precords = []
if psql != "":
    conn2 = pypyodbc.connect(
        connString,
        autocommit=True)
    print(connString)
    cursor = conn2.cursor()
    sqlcmd1 = psql
    print(sqlcmd1)
    cursor.execute(sqlcmd1)
    while True:
        dbrow = cursor.fetchone()
        if not dbrow:
            break
        row = ProfessorModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[3],
dbrow[4], dbrow[5], dbrow[6],
                        dbrow[7])
        precords.append(row)
    return render_template('SQLResult.html', crecords=crecords, precords=precords)

```

```

@app.route('/TextToGraph')
def TextToGraph():
    global userID, userName
    return render_template('TextToGraph.html')

```

```

from nltk.corpus import stopwords

```

```

from nltk.tokenize import word_tokenize
import pandas as pd

@app.route("/ProcessTextToGraph",methods = ['POST'])
def ProcessTextToGraph():
    global userName, userID
    etext = request.form['etext']
    etext = etext.lower()
    stop_words = set(stopwords.words('english'))
    stop_words.discard("all")
    print(type(stop_words), stop_words)
    word_tokens = word_tokenize(etext)

    filtered_sentence = [w for w in word_tokens if not w in stop_words]

    filtered_sentence = []

    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)

    print(word_tokens)
    print(filtered_sentence)
    csql = ""
    psql = ""
    tsql = ""
    print(etext)
    csql = "SELECT * FROM classroom"
    psql = "SELECT * FROM professor "
    print(csql)
    crecords = []
    if csql != "":
        conn2 = pypyodbc.connect(
            connString,
            autocommit=True)
        print(connString)
        cursor = conn2.cursor()
        sqlcmd1 = csql
        print(sqlcmd1)

    SQL_Query = pd.read_sql_query(csql, conn2)
    crecords = pd.DataFrame(SQL_Query)

    ""
    while True:
        dbrow = cursor.fetchone()
        if not dbrow:

```

```

        break
        row = ClassroomModel(dbrow[0], dbrow[1], dbrow[2], dbrow[3], dbrow[3],
dbrow[4], dbrow[5], dbrow[6],
                        dbrow[7], dbrow[8], dbrow[9])
        crecords.append(row)
    ""
print(psql)
precords = []
if psql != "":
    conn2 = pypyodbc.connect(
        connString,
        autocommit=True)
    print(connString)
    cursor = conn2.cursor()
    sqlcmd1 = psql
    print(sqlcmd1)
    SQL_Query = pd.read_sql_query(psql, conn2)
    precords = pd.DataFrame(SQL_Query)
import base64
filename = "";
if "bar" in etext and "classroom" in etext:
    print(etext)
    ax = sns.barplot(x="classroomname", y="noofseats", data=crecords)
    if "seat" in etext or "seats" in etext:
        ax = sns.barplot(x="classroomname", y="noofseats", data=crecords,
hue="hue")
        if "building name" in etext or "building" in etext:
            ax = sns.barplot(x="buildingname", y="noofseats", data=crecords,
hue="hue")
            if "block" in etext or "blocks" in etext:
                ax = sns.barplot(x="block", y="noofseats", data=crecords, hue="hue")
            if "facility1" in etext or "facility 1" in etext:
                ax = sns.barplot(x="facility1", y="noofseats", data=crecords, hue="hue")
        if "computer" in etext or "computers" in etext:
            ax = sns.barplot(x="classroomname", y="noofcomputers", data=crecords,
hue="hue")
            if "building name" in etext or "building" in etext:
                ax = sns.barplot(x="buildingname", y="noofcomputers", data=crecords,
hue="hue")
            if "block" in etext or "blocks" in etext:
                ax = sns.barplot(x="block", y="noofcomputers", data=crecords, hue="hue")
            if "facility1" in etext or "facility 1" in etext:
                ax = sns.barplot(x="facility1", y="noofcomputers", data=crecords,
hue="hue")
    ax.legend_.remove()
    fig = ax.get_figure()
    unqid = str(uuid.uuid1())

```


B. SCREENSHOTS

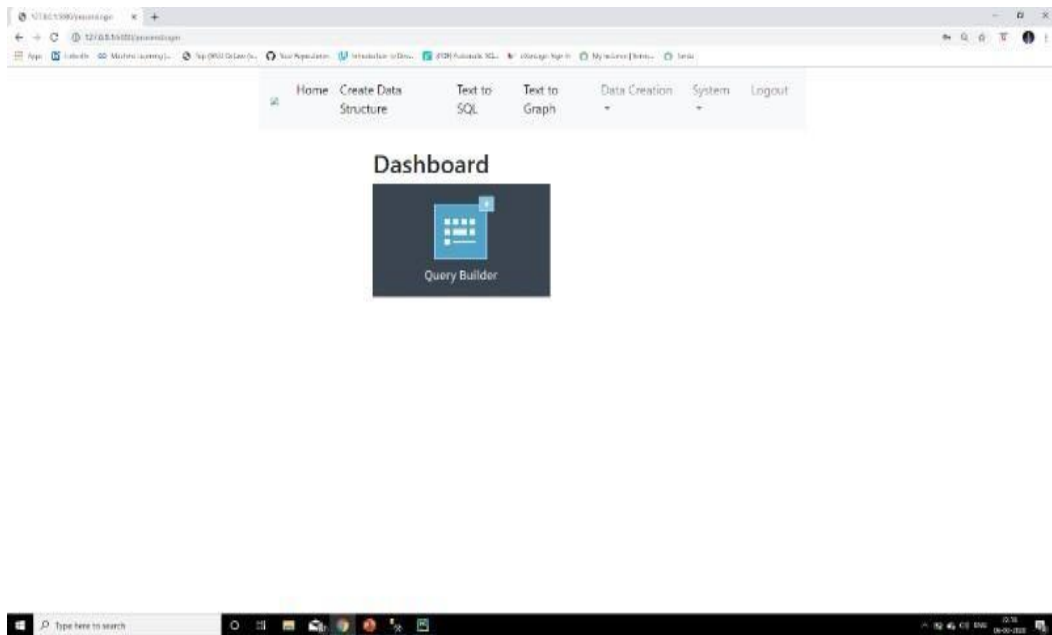


Fig B.1 Home Page

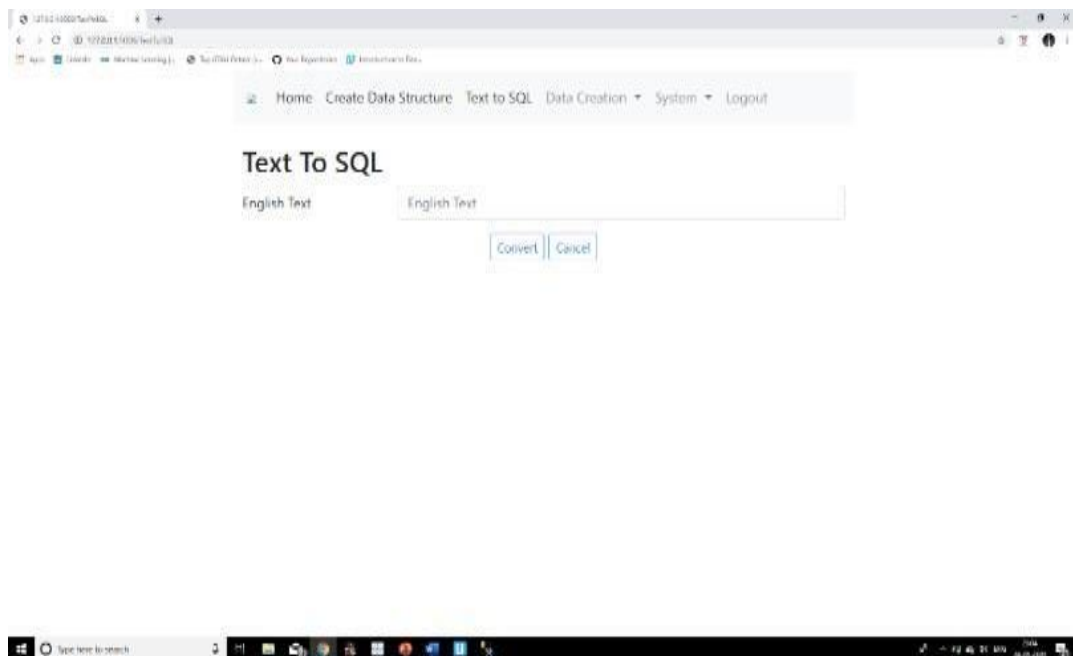


Fig B.2 Text Input Page

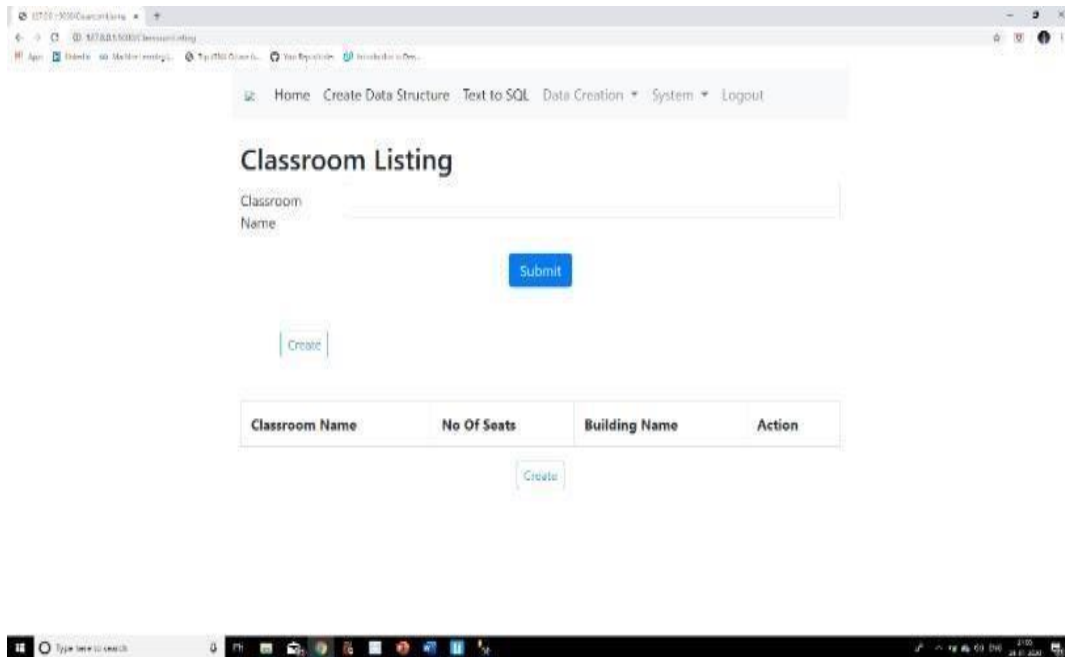


Fig B.3 Creating Database

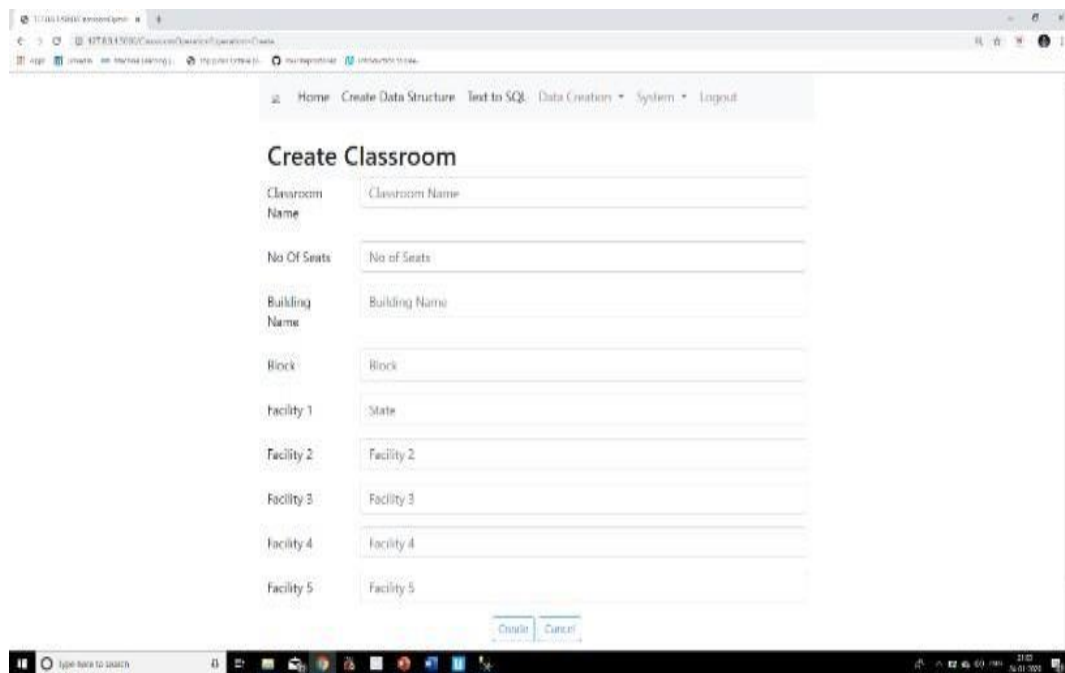


Fig B.4 Database Input

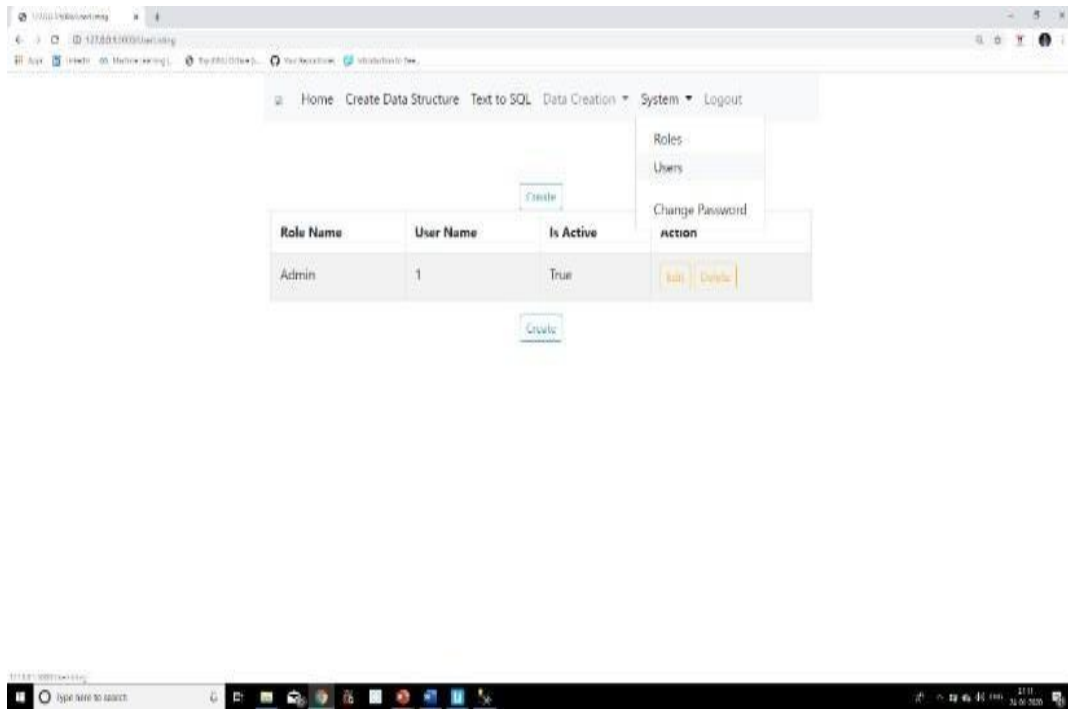


Fig B.5 List of Roles



International Journal for Innovative Engineering and Management Research

A Peer Reviewed Open Access International Journal

www.ijiemr.org

COPY RIGHT

2020 IJIEMR. Personal use of this material is permitted. Permission from IJIEMR must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. No Reprint should be done to this paper, all copy right is authenticated to Paper Authors

IJIEMR Transactions, online available on 27th Apr 2020. Link

[:http://www.ijiemr.org/downloads.php?vol=Volume-09&issue=ISSUE-04](http://www.ijiemr.org/downloads.php?vol=Volume-09&issue=ISSUE-04)

Title: **QUERY GENERATION USING NADAQ SYSTEM**

Volume 09, Issue 04, Pages: 140-145

Paper Authors

MR.CHANDRA RAJA.P, MR.PAVAN SANTHOSH.I, MRS.DIVYA.S



USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per **UGC Guidelines** We Are Providing A Electronic Bar Code

QUERY GENERATION USING NADAQ SYSTEM

MR.CHANDRA RAJA.P[1],MR.PAVAN SANTHOSH.I[2],MRS.DIVYA.S [3]

^{1,2}UG Student, Dept. Of Computer Science, Sathyabama University, Chennai,India

³Assistant professor, Dept. of computer Science, Sathyabama University, Chennai,India

Abstract—Natural Language Generation (NLG) utilizes Artificial Intelligence (AI) and Natural Language Processing (NLP) procedures to construct computational frameworks that create reports, stories and different sorts of writings in human languages. NLG utilizes examination, AI, and NLP to acquire pertinent data about non-etymological information and to create literary synopses and clarifications of these information which assist individuals with comprehension and advantage from them. Right now, is an exploration field that tends to the information esteem chain by utilizing natural language as a device for conquering any hindrance between crude information and important data imparted to clients in an understandable manner, adjusted to their data needs. The proposed framework train start to finish neural system models comprising of an encoder model to deliver a concealed portrayal of the source content, trailed by a decoder model to create the objective. While such models have altogether less pieces than prior frameworks, critical tuning is as yet required to accomplish great execution. For content generation models specifically, the decoder can act in undesired ways, for example, by creating shortened or dull yields, yielding tasteless and nonexclusive reactions, or now and again delivering ungrammatical jabber. This paper is expected as a functional guide for settling such undesired conduct in content generation models, with the point of empowering genuine applications.

Keywords—Neural Network, Natural Language Processing, Term Frequency

I.INTRODUCTION

Information sharing among different associations could assist with encouraging the proof based treatment by consolidating confirmations from heterogeneous medical clinics datasets Healthcare scientists and clinicians expect apparatuses to remove pertinent data from clinical data framework's information (2018). These appropriated databases contain various information configuration models e.g., Entity Relationship and Entity Attribute Value. Safari et al. proposed a calculation to interpret Restricted Natural Language Query

to SQL (2014). Conventional algorithms have been utilized for mapping and interpretation. In the initial step question terms are mapped to Restricted Natural Language Query by means of (Clinical Data Analytic Language) interface. Next fleeting articulation of the inquiry is deciphered by means of a 2-layer rule based procedure. Interpretation from Restricted Natural Language Query to SQL is performed by means of Top-K calculation on the base of comparability that is additionally used by for the mapping procedure. The executed model was tried on four classifications of questions

and it accomplished 84% of precision. Li et al. displayed a methodology which manages complex info questions of numerous spaces to make an interpretation of them into SQL inquiries in a nonexclusive manner (Li and Jagadish, 2014). The subsequent SQL articulations incorporate question settling, inquiry joins, and inquiry total. A framework has been created dependent on the proposed approach named as NaLIR (Natural Language Interface for Relational databases), which fuse these qualities. The framework reuses past SQL explanations from the inquiry log to spare question calculation time. TiQi, a characteristic language interface, permits to present discourse and content based inquiries in common language (Lin, 2015). It is an online apparatus and particularly intended to get to extend's information. TiQi acknowledges client question and creates Traceability Information Model (TIM) which shows hidden item classes and properties. TIM is put away in a brought together area to outline hubs to get to and indicate information requested by input inquiry. So as to deliver an exceptional SQL yield, H2, the JAVA SQL database, has been planned. This database motor offers help for information sources running from Jira to Excel Spreadsheet.

Term frequency-inverse document frequency depends on Bow, yet gives more detail than essentially taking term frequency, as was done in the earlier model. TFIDF yields a worth that shows how significant a given word is by taking a gander at term frequency, yet in addition investigating how often the word shows up over all documents.

The primary bit, term frequency, is moderately direct.

$$\text{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

condition ascertains log of absolute number documents in our corpus, isolated by all the documents in which the term that we are assessing shows up. In our code, we ascertain this with the capacity (2). Presently, we are prepared to continue to the last advance of the calculation, which is increasing the term frequency by the inverse document frequency, as appeared in the first code.

II. CORE CONCEPTS

The term neural network, as it is typically utilized, is really a misnomer. PCs endeavor to reenact organic neural networks by executing counterfeit neural networks. In any case, most productions utilize the expression "neural network," instead of "counterfeit neural network" (ANN). This book takes action accordingly. Except if the expression "neural network" is unequivocally prefixed with the expressions "organic" or "counterfeit" you can accept that the expression "fake neural network" is expected. To investigate this qualification, you will initially be demonstrated the structure of an organic neural network. To develop a PC equipped for "human-like idea," analysts have utilized the main working model they have accessible—the human mind. Nonetheless, the human cerebrum overall is awfully intricate to show. Or maybe, the individual cells that make up the human mind are considered. At

the most essential level, the human mind is made principally out of neuron cells. They are the fundamental structure squares of the human cerebrum. Fake neural networks endeavor to reproduce the conduct of these cells. Natural neural networks are simple. As you will find in the following area, reenacting simple neural networks on an advanced PC can show a few difficulties. Neurons acknowledge a simple sign through their dendrites. Since this sign is simple, the voltage of each sign will shift. On the off chance that the voltage is inside a specific range, the neuron will fire. At the point when a neuron fires, another simple sign is transmitted from the terminating neuron to different neurons. This sign is led over the terminating neuron's axon. The area so information and yield are recalled neural connections. Afterward, in section 5, The Feedforward Backpropagation Neural Network model will show that the neurotransmitters are the interface between a program and a neural network. A neuron settles on a choice by terminating or not terminating. The choices being settled on are very low-level choices. It requires an enormous number of choices to be made by numerous neurons just to peruse this sentence. More significant level choices are the consequence of the aggregate information and yield of numerous neurons. The individual neurons that make up a neural network are interconnected through their neurotransmitters. These associations permit the neurons to flag each other as data is handled. Not all associations are equivalent. Every association is allotted an association weight. On the off chance that

there is no association between two neurons, at that point their association weight is zero. These loads are what decide the yield of the neural network; accordingly, it very well may be said that the association loads structure the memory of the neural network.

A. Preparing Neural Networks

Preparing is the procedure by which these association loads are doled out. Most preparing algorithms start by doling out irregular numbers to load slatice. At that point, the legitimacy of the neural network is analyzed. Next, the loads are balanced dependent on how well the neural network performed and the legitimacy of the outcomes. This procedure is hashed until the approval mistake is inside a satisfactory point of confinement. There are numerous approaches to prepare neural networks. Neural network preparing strategies by and large fall into the classes of regulated, solo, and different half and half methodologies.

Administered preparing is cultivated by giving the neural network a lot of test information alongside the foreseen yields from every one of these examples. Administered preparing is the most well-known type of neural network preparing. As administered preparing continues, the neural network is taken through various emphases, or ages, until they yield of the neural network coordinates the foreseen yield, with a sensibly little pace of mistake. Every age is one go through the preparation tests.

Unaided preparing is like directed preparing, then again, actually no foreseen yields are given. Solo preparing for the most part

happens when the neural network is being utilized to arrange contributions to a few gatherings. The preparation includes numerous ages, similarly as in directed preparing. As the preparation advances, the arrangement bunches are "found" by the neural network..

B. Approving Neural Networks

A neural system is an interconnected get together of straightforward preparing components, units or hubs, whose usefulness is approximately founded on the creature neuron.

The handling capacity of the system is put away in the interunit association qualities, or loads, acquired by a procedure of adjustment to, or gaining from, a lot of preparing designs. To substance this out a little we first investigate some essential neurobiology. The human mind comprises of an expected 10¹¹ (100 billion) nerve cells or neurons, an exceptionally adapted case of which is appeared in Figure 1.1. Neurons impart by means of electrical signs that are brief motivations or "spikes" in the voltage of the cell divider or layer. The interneuron associations are interceded by electrochemical intersections called neural connections, which are situated on parts of the phone alluded to as dendrites.

$$\hat{g}_0 = \frac{1}{N} \sum_{n=0}^{N-1} w_N^{-n0} g_n = \frac{1}{N} \sum_{n=0}^{N-1} g_n.$$

Every neuron ordinarily gets a large number of associations from different neurons and in this manner continually getting a huge number of approaching signs, which in the long run arrive at the cell body. Here, they are incorporated or added

together somehow or another and, generally, on the off chance that the subsequent sign surpasses some limit, at that point the neuron will "fire" or produce a voltage motivation accordingly. This is then transmitted to different neurons by means of a fanning fiber known as the axon. In deciding if a motivation ought to be created or not, some approaching signs produce an inhibitory impact and will generally forestall terminating, while others are excitatory and advance drive age. The unmistakable handling capacity of every neuron is then expected to live in the sort—excitatory or inhibitory—and quality of its synaptic associations with different neurons. It is this design and style of preparing that we want to join in neural systems and, on account of the accentuation on the significance of the interneuron associations, this kind of framework is once in a while alluded to as being connectionist and the investigation of this general methodology as connectionism. This wording is regularly the one experienced for neural systems with regards to mentally enlivened models of human psychological capacity. Nonetheless, we will utilize it for the most part to allude to neural systems without reference to a specific field of use.

A mainstream worldview of learning, called learning with an instructor, or managed learning, includes adjustment of the synaptic loads of a neural system by applying a lot of marked preparing models, or undertaking models. Every model comprises of a one of a kind information signal and a comparing wanted (target) response. The organize is given a model picked aimlessly

from the set, and the synaptic loads (free parameters) of the system are altered to limit the contrast between the ideal reaction and the real reaction of the system delivered by the info signal as per a suitable factual foundation. The preparation of the system is rehased for some models in the set, until the system arrives at a consistent state where there are no further noteworthy changes in the synaptic weights. The recently applied preparing models might be reapplied during the instructional meeting, yet in an alternate order.

$$\hat{g}(v\Delta k) = \int_{-\infty}^{\infty} g(x) \exp(-2\pi i v \Delta k x) dx$$

The expression "nonparametric" is utilized here to imply the way that no earlier suppositions are made on a factual model for the information. Consider, for instance, an example grouping task, where the prerequisite is to allocate an information signal speaking to a physical item or occasion to one of a few prespecified classifications (classes). In a nonparametric way to deal with this issue, the necessity is to "gauge" self-assertive choice limits in the information signal space for the example characterization task utilizing a lot of models, and to do so without summoning a probabilistic conveyance model. A comparable perspective is understood in the administered learning worldview, which recommends a nearby relationship between the info yield mapping performed by a neural system and nonparametric measurable deduction.

C. Characterization

Characterization is the way toward ordering contribution to gatherings. For instance, an insurance agency might need to order protection applications into various hazard classifications, or an online association may need its email framework to arrange approaching mail into gathering of spam and non-spam messages. Frequently, the neural network is prepared by giving it an example gathering of information and direct ions regarding which bunch every datum component has a place. This permits the neural network to get familiar with the attributes that may show bunch enrollment.

D. Expectation

Expectation is another basic application for neural networks.

Given a time sensitive arrangement of information, a neural network will anticipate future qualities. The precision of the estimate will be reliant upon numerous variables, for example, the amount and significance of the information. For instance, neural networks are ordinarily applied to issues including foreseeing developments in monetary markets.

E. Pattern Recognition

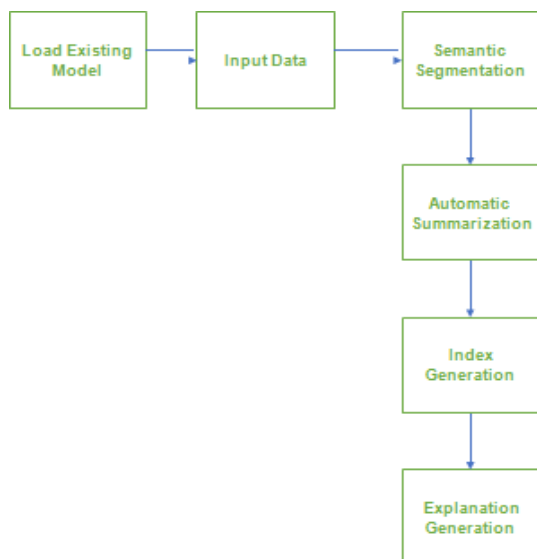
Pattern Recognition is one of the most well-known utilizations for neural networks. Example acknowledgment is a type of grouping. Example acknowledgment is essentially the capacity to perceive an example. The example must be perceived in any event, when it is misshaped. Think about the accompanying ordinary utilization of example acknowledgment. Each individual who holds a driver's permit ought to have the option to precisely distinguish a

traffic light. This is an incredibly basic example acknowledgment methodology did by endless drivers consistently. Be that as it may, few out of every odd traffic light appears to be identical, and the presence of a specific traffic light can be adjusted relying upon the hour of day or the season. Furthermore, numerous varieties of the traffic light exist. In any case, perceiving a traffic light is anything but a hard assignment for a human driver.

III. PROPOSED SYSTEM

Early, begin format the paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

A. Architecture



B. Units

- NLP based Explanation
- Automatic Generation
- High-quality results
- Support for both correct prediction and wrong prediction

- Can work for large datasets

IV. CONCLUSION

The two fundamental qualities is a fair, implying can comprise additional backup reasons why the forecast ought to be valid, or an elective explanation for the conflicting expectation—these attributes give clients more data identified with the forecast so they can settle on a progressively educated choice. we performed careful factual tests indicating that the selection of loads forth equality capacity, included determination method, and grouping calculation all have measurably critical impacts on the general nature of the clarifications got.

V. REFERENCES

- [1] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier." in Proc KDD, 2016, pp. 1135-1144.
- [2] M. Anthony and J. Ratsaby, "A hybrid classifier based on boxes and nearest neighbors. Discrete Appl. Math., 172. pp. 1-11. Jul 2014
- [3] P. Serafini, "Classifying negative and positive points by optimal 1 box clustering," Discrete Appl. Math., vol. 165, pp. 270-282. Mar. 2014.
- [4] M. Kirmse and U. Petersohn, "Large margin Neurocomputing, vol. 130, pp. 53-62. Apr. 2014
- [5] S. Kumar, E. Seera, "Metric program explanations for Complex Separator function" Proc. SUM, logie 2016.. pp. 199-213
- [6] Henelius et al, "Goldeneye: A closer look into the in Proc. SLDS, 2015. pp. 96-105



International Journal for Innovative Engineering and Management Research

A Peer Reviewed Open Access International Journal

www.ijiemr.org