

A
Project Report
On
**ABSTRACTIVE TRANSCRIPT SUMMARIZATION OF YOUTUBE
VIDEOS**

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

Bachelor of Technology
in
COMPUTER SCIENCE AND ENGINEERING

By
P. Sri Charan
(20EG105338)

L. Sree Vidhya
(20EG105347)

K. Shreya Reddy
(20EG105357)

Under the guidance of

Mr. G. Kiran Kumar
Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Venkatapur (V), Ghatkesar (M), Medchal (D), T.S - 500088
(2023-2024)



SCHOOL OF
ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled “**Abstractive Transcript Summarization of YouTube Videos**” being submitted by **P. Sri Charan** bearing the Hall Ticket number **20EG105338**, **L. Sree Vidhya** bearing the Hall Ticket number **20EG105347**, **K. Shreya Reddy** bearing the Hall Ticket number **20EG105357** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** in **Anurag University** is a record of bonafide work carried out by them under my guidance and supervision in academic year 2023 to 2024.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Internal Guide

Mr. G. Kiran Kumar

Assistant Professor

Dean, CSE

Dr. G. Vishnu Murthy

External Examiner

DECLARATION

We hereby declare that the report entitled “**Abstractive Transcript Summarization of YouTube Videos**” submitted to **Anurag University** in partial fulfilment of the requirements for the award of the degree **Bachelor of Technology (B. Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Mr. G. Kiran Kumar, Assistant Professor** and this project work have not been submitted to any other University or Institution for the award of any other degree or diploma.

Place: Anurag University, Hyderabad

Date:

P. Sri Charan
(20EG105338)

L. Sree Vidhya
(20EG105347)

K. Shreya Reddy
(20EG105357)

ACKNOWLEDGMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Mr. G. Kiran Kumar, Assistant Professor, Department of Computer Science and Engineering**, Anurag University for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar, Professor & Dean, School of Engineering**, Anurag University, for their encouragement and timely support in our B.Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy, Dean, Department of Computer Science and Engineering**, Anurag University.

We also express our deep sense of gratitude to **Dr. V. V. S. S. Balram**, Academic Co-ordinator, **Dr. T. Shyam Prasad**, Assistant Professor, Project Co-ordinator and Project Review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

P. Sri Charan
(20EG105338)

L. Sree Vidhya
(20EG105347)

K. Shreya Reddy
(20EG105357)

ABSTRACT

The abundance of videos on YouTube makes it challenging to keep up with the vast amount of information available on a daily basis. Watching every video in full requires significant time, and many users seek faster ways to grasp the key points without having to watch the entire content. By leveraging advanced natural language processing models like Seq2Seq RNN-LSTM and PEGASUS, the project provides an innovative approach to generating summaries of YouTube videos. These models use neural networks to analyze the input data and extract essential information, resulting in concise summaries. To enhance the efficiency and accuracy of the summaries, the project combines multiple models instead of relying solely on one. This approach allows for a comprehensive examination of the input data, resulting in more precise summaries that better capture the nuances of the content. Additionally, the implementation of BERT, a powerful model known for its superior contextual understanding, contributes to the creation of high-quality summaries. Let's consider a scenario where a student is preparing for an exam and has access to numerous YouTube videos related to the subject matter. The student's time is limited, and watching each video in its entirety is not feasible. With the proposed system, the student can input the URLs of the relevant videos, and the system will efficiently generate concise summaries of each video. These summaries highlight the most important points, enabling the student to absorb the essential information quickly and effectively. As a result, the student can better manage their study time, focus on key concepts, and improve their exam preparation.

Keywords: Deep Learning, YouTube Video Summarization, Seq2Seq RNN-LSTM, PEGASUS, Ensemble Method, BERT.

TABLE OF CONTENTS

| SNO | CONTENT | PAGE NO |
|-----|---|---------|
| 1 | Introduction | 1 |
| | 1.1 Objective | 1 |
| | 1.2 Problem Statement | 1 |
| | 1.3 Software Requirements | 2 |
| | 1.4 Hardware Requirements | 3 |
| 2 | Literature Survey | 5 |
| | 2.1 Comparison of Existing Strategies for Problem Solve | 5 |
| | 2.2 Comparison of Existing Method for Selected Strategies | 6 |
| 3 | System Analysis | 8 |
| | 3.1 Existing System | 8 |
| | 3.1.1 Disadvantages of Existing System | 8 |
| | 3.2 Proposed Method | 9 |
| | 3.2.1 Advantages of Proposed Method | 9 |
| | 3.3 Functional Requirements | 10 |
| | 3.4 Non-Functional Requirements | 10 |
| | 3.5 Concept Tree | 11 |
| 4 | System Design | 12 |
| | 4.1 System Architecture | 12 |
| | 4.2 Data Flow Diagram | 13 |
| 5 | Implementation | 15 |
| | 5.1 List of Program Files | 15 |
| | 5.2 Modules | 15 |
| | 5.3 Algorithms Used | 16 |
| | 5.3.1 BERT Model | 16 |
| | 5.3.2 Seq2Seq RNN-LSTM | 16 |
| | 5.3.3 PEGASUS | 17 |
| | 5.4 Packages Used | 18 |
| | 5.5 Attributes | 21 |

| | | |
|----|--|----|
| | 5.6 Datasets | 23 |
| | 5.6.1 Overview | 23 |
| | 5.6.2 Structure | 23 |
| | 5.6.3 Characteristics | 23 |
| | 5.6.4 Usecases | 24 |
| | 5.6.5 Availability | 24 |
| | 5.6.6 Impact | 24 |
| | 5.7 Sample Code | 25 |
| 6 | Experiment Results | 34 |
| 7 | Experimental Setup | 37 |
| | 7.1 Software Environment | 37 |
| | 7.1.1 Installation of Anaconda for Python | 37 |
| | 7.1.2 Python Anaconda Installation | 38 |
| | 7.2 Python Language | 44 |
| | 7.2.1 Features of Python | 44 |
| 8 | Parameter Formulas | 46 |
| | 8.1 Parameter Comparison Table | 48 |
| 9 | Comparison of Results | 49 |
| | 9.1 Comparative Analysis of Summarization Techniques | 49 |
| | 9.2 Impact of Ensemble Approach | 50 |
| | 9.3 Generalization to Noisy Scenarios | 50 |
| 10 | Discussion of Results | 51 |
| 11 | Conclusion | 52 |
| 12 | References | 53 |

LIST OF FIGURES

| SNO | FIG NO | TITLE OF FIGURE | PAGE NO |
|-----|--------|--|---------|
| 1 | 3.1 | Concept Tree | 11 |
| 2 | 4.1 | System Architecture | 13 |
| 3 | 4.2 | Data Flow Diagram | 14 |
| 4 | 6.1 | Command Prompt | 34 |
| 5 | 6.2 | Landing Page | 34 |
| 6 | 6.3 | User Registration Page | 35 |
| 7 | 6.4 | Login Page | 35 |
| 8 | 6.5 | Dashboard | 36 |
| 9 | 6.6 | Result | 36 |
| 10 | 7.1 | Anaconda | 37 |
| 11 | 7.2 | Anaconda Installer | 38 |
| 12 | 7.3 | Anaconda Installation | 39 |
| 13 | 7.4 | Anaconda License Agreement | 39 |
| 14 | 7.5 | Anaconda Setup | 40 |
| 15 | 7.6 | Choose Destination Folder | 40 |
| 16 | 7.7 | Anaconda Advanced Installation Option | 41 |
| 17 | 7.8 | Installation of Anaconda | 41 |
| 18 | 7.9 | Installation Complete | 42 |
| 19 | 7.10 | Window about PyCharm | 42 |
| 20 | 7.11 | Installation Complete Window | 43 |
| 21 | 7.12 | Anaconda Prompt | 43 |
| 22 | 9.1 | Comparison of Evaluation of Metrics between TF-IDF and Ensemble Model | 49 |
| 23 | 9.2 | Precision-Recall Curve | 49 |
| 24 | 9.3 | Comparison of Evaluation Metrics between Individual Models and Ensemble Model | 50 |
| 25 | 9.4 | Impact of Noise Level of Model Performance | 50 |

LIST OF TABLES

| SNO | TABLE NO | TITLE OF THE TABLE | PAGE NO |
|-----|----------|---|---------|
| 1 | 2.1 | Comparison of Existing Strategies for Problem Solving | 5 |
| 2 | 2.2 | Comparison of Existing Method for Selected Strategy | 6 |
| 3 | 8.1 | Parameter Comparison Table | 48 |

1. Introduction

Since the advent of digital multimedia, a vast array of content has flooded social platforms daily, including news, documentaries, movies, talk shows, and sports events. This abundance of diverse content demands significant processing time and memory. To address this, condensing videos becomes crucial, enabling consumers to obtain maximum information in minimal time. Watching videos is passive, unlike reading, making it challenging to absorb all information actively. Hence, condensing video content becomes essential for effective consumption. Artificial text summarization technologies offer a solution by extracting pertinent details from the video more efficiently. This not only reduces the volume of video data but also facilitates quicker navigation. Users can glance at summaries of YouTube videos to determine if the content aligns with their interests, potentially saving them time.

1.1 Objective:

This study aims to advance abstractive transcript summarization for YouTube videos by employing an ensemble approach, combining Seq2Seq RNN-LSTM, PEGASUS, and BERT for enhanced accuracy and robustness. The investigation builds upon a base paper, extending the use of ensemble methods and introducing BERT to optimize summarization efficacy, ultimately refining the extraction of key information from diverse video sources.

1.2 Problem Statement:

The study addresses the challenge of improving abstractive transcript summarization for YouTube videos. Despite existing methods like Seq2Seq LSTM and PEGASUS, there's a need for enhanced accuracy and robustness. The investigation explores an ensemble approach, incorporating BERT, to optimize performance and extract key information effectively from diverse video sources, aiming to refine summarization in the dynamic realm of YouTube content.

1.3 Software Requirements:

Software requirements outline the resources and prerequisites necessary for optimal application functionality. These elements, typically not bundled with the installation package, must be separately installed prior to software installation.

Platform: In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

Software requirements often prioritize the operating system. Compatibility issues arise between different versions of the same operating system line. For instance, software for Windows XP may not work on Windows 98, though the reverse can sometimes hold true. Similarly, software designed for newer features of Linux Kernel v2.6 may not function properly on distributions using Kernel v2.2 or v2.4.

APIs and drivers: Software relying on specialized hardware, such as high-end display adapters, requires specific APIs or updated device drivers. DirectX, for example, offers APIs for multimedia tasks, particularly game programming, on Microsoft platforms.

Web browser: Many web applications and software reliant on Internet technologies utilize the system's default browser. Microsoft Internet Explorer, often chosen for Windows, utilizes ActiveX controls despite their vulnerabilities.

1) Software: Anaconda

2) Primary Language: Python

3) Frontend Framework: Flask

4) Back-end Framework: Jupyter Notebook

5) Database: Sqlite3

6) Front-End Technologies: HTML, CSS, JavaScript and Bootstrap4

1.4 Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

Architecture: All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power: The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory: All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage: Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter: Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals: Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

1)Operating System: Windows Only

2)Processor: i5 and above

3)RAM: 8GB and above

4)Hard Disk: 256 GB in local drive

2. Literature Survey

In recent advancements in natural language processing and video summarization, researchers have explored various strategies for efficiently summarizing video content [1]. One approach involves using attentive neural networks for abstractive sentence summarization, which leverage LSTM and SVO models to capture temporal relationships within the video data [2]. This method is known for its simple implementation and efficiency; however, it can produce disjointed summaries if crucial information is missing or erroneous, and it may struggle with lengthy videos. Another notable approach is automatic summarization using methods such as Multistage Fusion Network with Forget Gate (MFFG), Single-Stage Fusion Network with Forget Gate (SFFG), and TF-IDF to extract keywords from YouTube videos [1]. This approach enables the quick delivery of essential information, effectively condensing lengthy videos into concise textual summaries to save time for users. Despite these benefits, challenges include capturing nuanced details across diverse content and the complexity involved in training these models, particularly the computational overhead due to the complex fusion approach in MFFG. Fine-tuning also requires high-quality data, adding to the overall challenge.

2.1 Comparison of Existing Strategies for Problem Solve:

| Sl.No | Strategies | Advantages | Disadvantages |
|-------|--|---|--|
| 1 | Abstractive sentence summarization with attentive neural networks. | It captures temporal relationships by using LSTM and SVO models. Works efficiently for simple implementation. | Result in disjointed descriptions if information is missing or erroneous. It cannot give an accurate summary for lengthy videos. |

| | | | |
|---|--|--|--|
| 2 | Automatic summarization, utilizing MFFG, SFFG and TF-IDF to extract keywords from YouTube videos. Quick delivery of essential information from lengthy videos. | Efficiently condenses lengthy YouTube videos into concise textual summaries, saving time for users with constraints. | Poses a challenge in capturing the nuanced details of diverse information. Complexity in training the models and the need for high-quality data for fine-tuning. Possible computational overhead due to the complex fusion approach, particularly in MFFG. |
|---|--|--|--|

2.2 Comparison of Existing Methods from Selected Strategies:

| Sl.No | Author | Strategies | Advantages | Disadvantages |
|-------|---|---|---|--|
| 1 | Aniqa Dilawari, Muhammma d Usman Ghani Khan . | Abstractive sentence summarization with attentive neural networks | It captures temporal relationships by using LSTM and SVO models. Works efficiently for simple implementation. | Result in disjointed descriptions if information is missing or erroneous. It cannot give an accurate summary for lengthy videos. |

| | | | | |
|---|---|---|--|---|
| 2 | Nayu Liu, Xian Sun, Hongfeng Yu, Fanglo ng Yao, Guang luan Xu and Kun Fu. | Automatic summarization, utilizing MFFG, SFFG and TF-IDF to extract keywords from YouTube videos. Quick delivery of essential information from lengthy videos. | Efficiently condenses lengthy YouTube videos into concise textual summaries, savin g time for users with constraints. | Poses a challenge in capturing the nuanced details of diverse informati on. Complexity in training the models and fine- tuning. Possible computational overhead due to the complex fusion approach, particularly in MFFG. |
|---|---|---|--|---|

3. System Analysis

3.1 Existing System:

The existing system leverages automatic summarization techniques, specifically tailored for YouTube videos, to efficiently distill key information from lengthy content. Employing natural language processing, the system transcribes and undergoes summary stages to generate concise textual summaries. The method relies on term frequency-inverse document frequency (TF-IDF) to extract vital keywords, considering both word and sentence counts. Addressing the challenge of time-consuming or dull videos, the system aims to deliver essential information in a condensed text format, particularly beneficial for students and researchers with time constraints. The primary goal is to facilitate quick access to valuable insights without the need to watch lengthy videos. Evaluation of the system's efficacy is conducted using the Rouge method on the CNN-dailymail-master dataset, ensuring the robustness and effectiveness of the existing approach in summarizing YouTube content.

3.1.1 Disadvantages of Existing System:

1. Relies on extractive summarization (TF-IDF), limiting the depth of summarization compared to the proposed abstractive methods.
2. May struggle with diverse content, as it primarily focuses on keyword extraction, potentially missing nuanced information.
3. The existing system may not adapt well to the evolving nature of YouTube content, leading to potential information gaps.
4. Sole reliance on the Rouge method and a specific dataset may not capture the system's adaptability to varied video genres.

3.2 Proposed Method:

This study focuses on advancing the abstractive summarization of YouTube videos, leveraging established techniques like Seq2Seq LSTMs and PEGASUS for their respective strengths in abstractive and extractive summarization. A key contribution is the introduction of an ensemble model, merging predictions from these models to enhance accuracy and resilience. Additionally, the study advocates for integrating BERT's capabilities into this ensemble for further performance gains. This novel approach signifies a significant leap in effectively summarizing YouTube content by not only building on existing methodologies but also combining their strengths synergistically. The ensemble model's comprehensive approach holds the potential to revolutionize video summarization, catering to the dynamic and diverse nature of content on the YouTube platform.

3.2.1 Advantages of Proposed Method:

1. Utilizes Seq2Seq LSTMs and PEGASUS for more nuanced and comprehensive summarization, allowing for better content understanding.
2. Integrates multiple models to enhance accuracy and resilience, addressing limitations of individual methods and improving overall summarization performance.
3. Harnesses BERT's capabilities for further performance gains, enabling a more sophisticated understanding of context and enhancing summarization accuracy.
4. The proposed ensemble approach has the potential to revolutionize video summarization by combining strengths, offering a more versatile solution for dynamic YouTube content.

3.3 Functional Requirements:

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modelling
5. Predicting

3.4 Non-Functional Requirements:

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement

- Maintainability requirement
- Regulatory requirement
- Environmental requirement

3.5 Concept Tree:

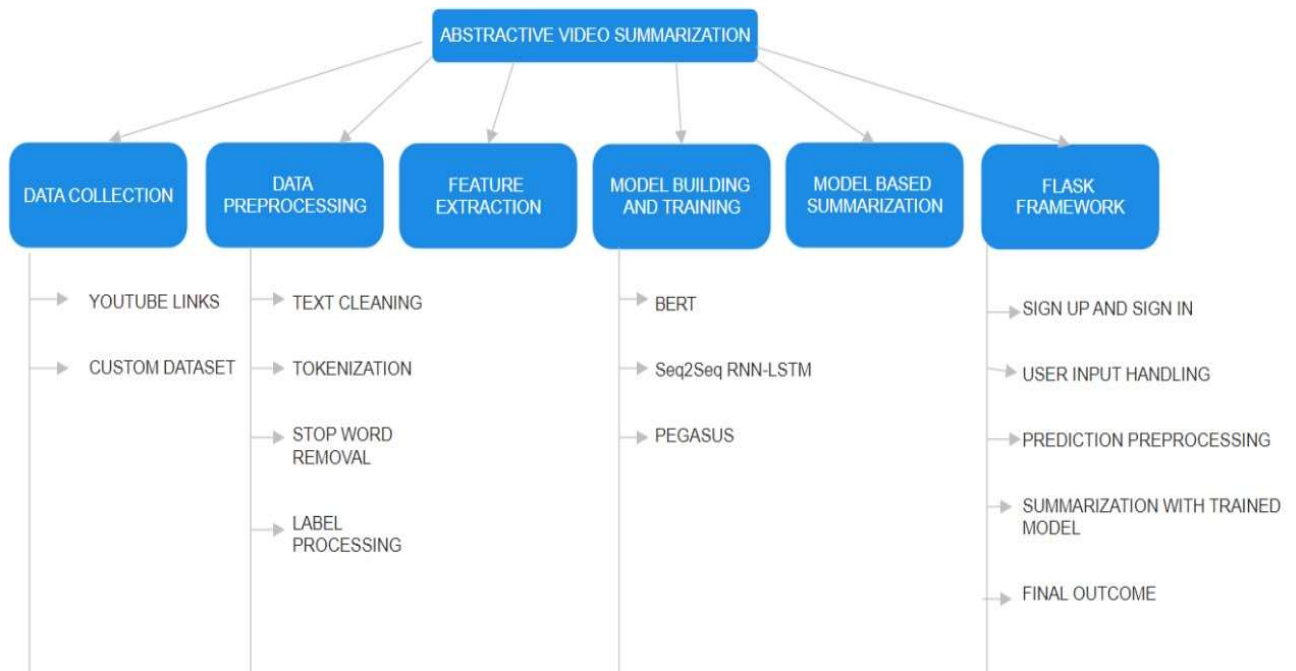


Figure 3.1: Concept Tree

4. System Design

4.1 System Architecture:

The data warehouse pipeline is a process which helps to prepare and analyze data for machine learning. By following these steps, one can build models that are more accurate and effective.

1) Data Acquisition: Data is collected from various sources, such as YouTube links, custom datasets, and news articles.

2) Data Preprocessing: The data is then cleaned and formatted for analysis. This may involve removing duplicates, handling missing values, and converting data types.

3) Exploratory Data Analysis (EDA): Exploratory data analysis is performed to get a better understanding of the data. This may involve calculating summary statistics, creating visualizations of the data, and identifying patterns and trends.

4) Feature Selection: Features are the independent variables that will be used to train the machine learning model. Feature selection is the process of selecting the most relevant features from the data.

5) Splitting the Data: The data is then split into two sets: a training set and a test set. The training set is used to train the machine learning model, and the test set is used to evaluate the performance of the model.

6) Model Building and Training: Here, different machine learning models are built and trained on the training data. Some of the models explored in this data warehouse include BERT, Seq2Seq RNN-LSTM, and PEGASUS.

7) Visualization: Once the models are trained, they are used to make predictions on the test data. The results are then visualized to understand how well the models performed.

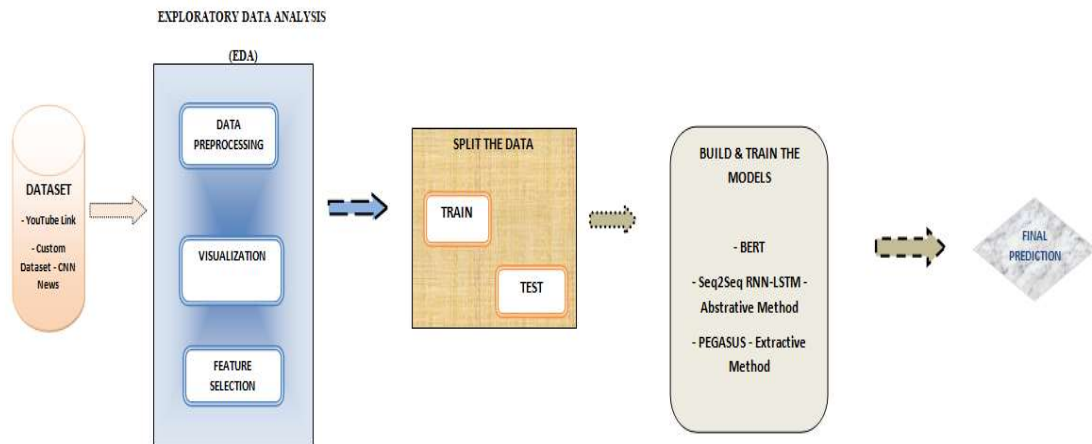


Figure 4.1: System Architecture

4.2 Data Flow Diagram:

- 1) The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- 2) The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- 3) DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- 4) DFD is also known as bubble chart. A DFD maybe used to represent a system at any level of abstraction. DFD maybe partitioned into levels that represent increasing information flow and functional detail.

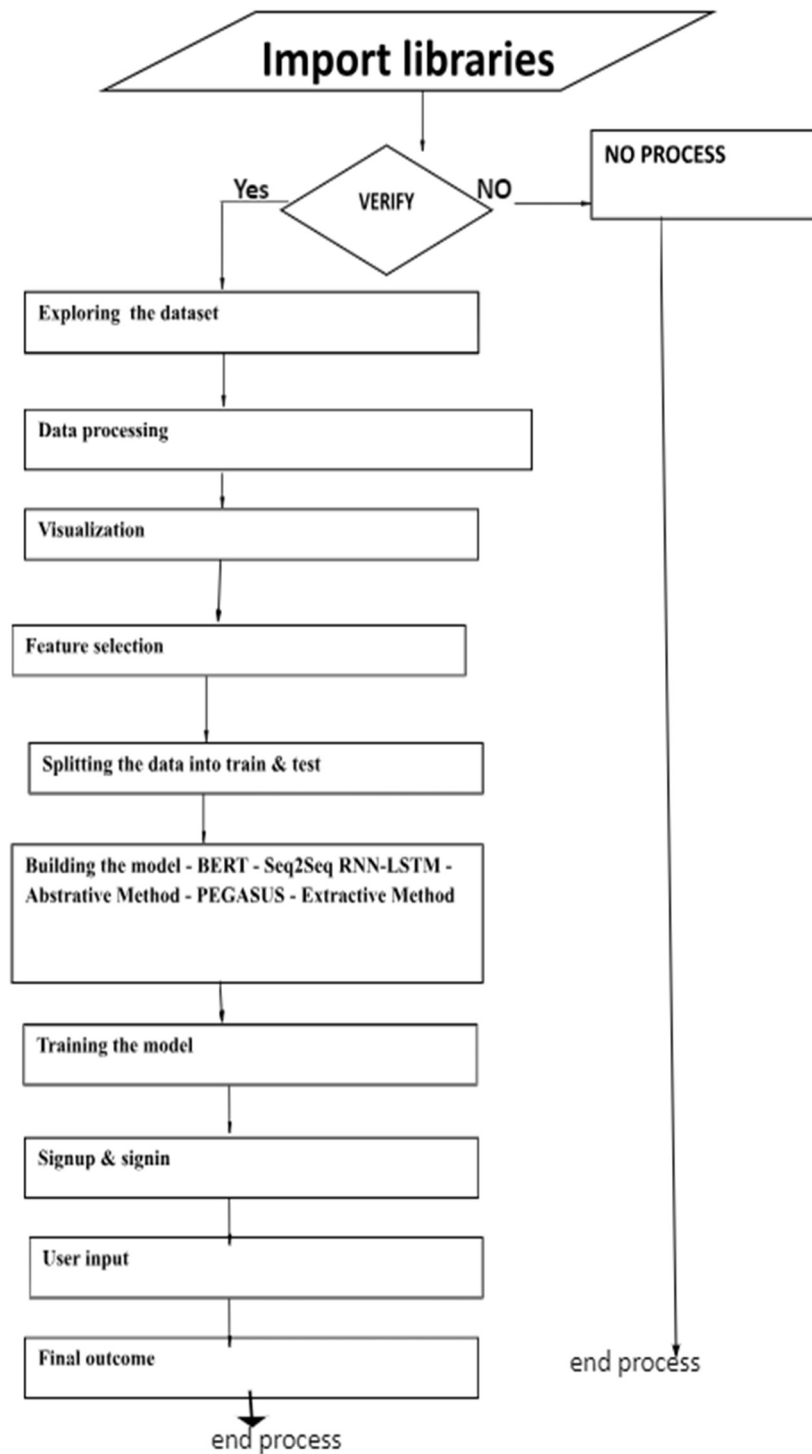


Figure 4.2: Data flow diagram

5. Implementation

5.1 List of Program Files:

- 1) about.html
- 2) home.html
- 3) index.html
- 4) result.html
- 5) signin.html
- 6) signup.html
- 7) notebook.html
- 8) val.html
- 9) app.py

5.2 Modules:

- 1) **Data loading:** using this module we are going to import the dataset.
- 2) **Data Preprocessing:** using this module we will explore the data.
- 3) **Splitting data into train & test:** using this module data will be divided into train & test.
- 4) **Model generation:** Model building - BERT - Seq2Seq RNN-LSTM - Abstractive Method - PEGASUS - Extractive Method. Algorithms accuracy calculated.
- 5) **User signup & login:** Using this module will get registration and login.

- 6) **User input:** Using this module will give input for prediction.
- 7) **Prediction:** final predicted displayed.

5.3 Algorithms Used:

5.3.1 Bidirectional Encoder Representations from Transformers (BERT):

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art natural language processing model developed by researchers at Google AI. Introduced in 2018, BERT is based on the transformer architecture, which efficiently processes sequences of text and captures contextual relationships. It uses a bidirectional approach, processing text in both directions to understand the context of a word based on its surroundings. BERT focuses on encoding input text into contextualized representations using a self-attention mechanism that prioritizes the most relevant words. In pre-training, BERT employs masked language modeling (MLM) to predict masked words in the context and next sentence prediction (NSP) to understand sentence relationships. After pre-training, BERT can be fine-tuned on specific downstream tasks such as sentiment analysis, question answering, and named entity recognition. BERT excels at capturing context, offers efficient transfer learning, and achieves top performance on various benchmarks and challenges, making it a foundational model for NLP tasks.

5.3.2 Seq2Seq RNN-LSTM (Sequence-to-Sequence Recurrent Neural Network with Long Short-Term Memory):

Seq2Seq RNN-LSTM (Sequence-to-Sequence Recurrent Neural Network with Long Short-Term Memory) is an abstractive summarization method widely used in natural language processing (NLP) tasks. It employs a neural network architecture composed of two main components: an encoder and a decoder, both based on RNN-LSTM. The encoder processes the input sequence (such as a sentence or document), capturing contextual relationships within the data and converting it into a fixed-size representation known as a context vector or thought vector. The decoder then uses this context vector to generate the output sequence, which in summarization tasks is the summary of the input sequence. The

decoder predicts one token (word) at a time based on the context vector and previously generated tokens. To enhance performance, particularly with long input sequences, many Seq2Seq models integrate an attention mechanism, which allows the decoder to focus on specific parts of the input sequence for more coherent and contextually accurate summaries. Seq2Seq RNN-LSTM offers nuanced content representation by producing summaries that are reinterpretations of the input rather than verbatim extracts, providing concise and flexible content representation. While the model's ability to handle varying input and output lengths makes it versatile for different NLP tasks, it can require substantial time and data for training, especially with longer input sequences. Additionally, without attention mechanisms, Seq2Seq models may struggle with long sequences as they risk losing context from earlier parts of the input. Despite these limitations, Seq2Seq RNN-LSTM remains a powerful approach to abstractive summarization, offering coherent and nuanced summaries of text data and demonstrating strong capabilities for a range of NLP applications.

5.3.3 PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization Sequence-to-sequence models):

PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization Sequence-to-sequence models) is a state-of-the-art natural language processing (NLP) model developed by Google AI for abstractive summarization. It leverages an advanced pre-training strategy called "Gap-sentences Generation" (GSG), in which full-text input documents have several sentences removed and replaced with special placeholder tokens, training the model to predict the missing sentences and comprehend the most crucial information from the text. Once pre-trained, PEGASUS can be fine-tuned on specific downstream summarization tasks, allowing it to adapt its summarization skills to different domains or styles and enhancing its performance. PEGASUS is known for its state-of-the-art performance, often surpassing other models in abstractive summarization, and produces high-quality, fluent summaries that capture the most important aspects of the original text. Additionally, it is versatile and can be adapted for other text generation tasks such as machine translation and text completion. However, training PEGASUS requires substantial computational resources and large datasets due to its complexity, and its sophisticated

architecture may be challenging for newcomers to NLP. Despite these limitations, the model's performance and adaptability make it a powerful tool for various text generation applications.

5.4 Packages Used:

- 1) **Flask**: Flask is a lightweight web application framework for Python. It allows developers to build web applications quickly and with minimal code. In this code, Flask is used to create web routes, render HTML templates, and handle HTTP requests.
- 2) **TensorFlow**: TensorFlow is a free open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.
- 3) **transformers**: Transformers is an open-source library provided by Hugging Face that provides implementations of state-of-the-art natural language processing (NLP) models. In this code, the Transformers package is used to load the T5 model for text summarization and its corresponding tokenizer.
- 4) **urllib**: The urllib module is a Python library that provides functions for working with URLs. In this code, the `urlparse` function from the `urllib.parse` submodule is used to extract the video ID from a YouTube URL.
- 5) **youtube_transcript_api**: This package is a Python library for retrieving transcripts and captions from YouTube videos. It provides a simple interface for accessing YouTube's transcript data.
- 6) **pandas**: Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was

majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

- 7) **Matplotlib:** Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB- like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.
- 8) **Scikit- learn:** Scikit- learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.
- 9) **numpy:** NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions. In this code, NumPy is used for numerical computations.
- 10) **pickle:** Pickle is a module in Python used for serializing and deserializing Python objects. It allows objects to be converted into a byte stream, which can then be stored in a file or transmitted over a network. In this code, pickle is used for saving and loading objects to and from files.

- 11) **sqlite3**: SQLite3 is a lightweight relational database management system included as a standard library in Python. It provides a simple way to create and interact with SQL databases. In this code, SQLite3 is used for storing user information in a database.
- 12) **random**: The random module is a Python library that provides functions for generating random numbers. In this code, the `randint` function from the random module is used to generate a random one-time password (OTP).
- 13) **smtplib**: The smtplib module is a Python library for sending emails using the Simple Mail Transfer Protocol (SMTP). It provides functions for connecting to an SMTP server and sending email messages. In this code, smtplib is used for sending OTP emails to users during the signup process.
- 14) **datetime**: The datetime module is a Python library that provides classes for manipulating dates and times. It allows for the creation, manipulation, and formatting of date and time objects. In this code, datetime is used for generating timestamps for email messages.

5.5 Attributes:

- 1) `app`: This is the Flask application object. It serves as the main entry point for the application and is used to register routes, handle requests, and manage other components such as templates and static files.
- 2) `model`: This attribute holds the T5 model instance from the Transformers library. It is used for text summarization tasks.
- 3) `tokenizer`: This attribute holds the T5 tokenizer instance from the Transformers library. It is used for tokenizing input text and decoding output from the T5 model.
- 4) `otp`: A global variable used to store the one-time password (OTP) for user verification during the signup process.
- 5) `username`, `name`, `email`, `number`, `password`: These are global variables used to store user information during the signup process.
- 6) `input_ids`: An attribute in the `summarizer` function, used to store the tokenized input text for the T5 model.
- 7) `summary_text`: An attribute in the `summarizer` function, used to store the decoded summary text generated by the T5 model.
- 8) `home()`: This function handles requests to the home route (`/`) and renders the 'home.html' template.
- 9) `about()`: This function handles requests to the 'about' route (`/about`) and renders the 'about.html' template.
- 10) `index()`: This function handles requests to the 'index' route (`/index`) and renders the 'index.html' template.

- 11) `extract_video_id(url)`: A utility function that extracts the video ID from a given YouTube URL.
- 12) `summarizer(script)`: A function that uses the T5 model to generate a summary of a given script. It encodes the input text, generates the summary using the model, and decodes the output.
- 13) `video_transcript()`: This function handles requests to the 'summarize' route (`/summarize`). It retrieves the video transcript using the YouTube Transcript API, generates a summary using the `summarizer` function, and renders the 'result.html' template with the summary and YouTube URL.
- 14) `logon()`: This function handles requests to the 'logon' route (`/logon`) and renders the 'signup.html' template.
- 15) `login()`: This function handles requests to the 'login' route (`/login`) and renders the 'signin.html' template.
- 16) `signup()`: This function handles requests to the 'signup' route (`/signup`). It retrieves user information from the request, generates an OTP, sends an email with the OTP, and renders the 'val.html' template for OTP validation.
- 17) `predict_lo()`: This function handles POST requests to the 'predict_lo' route (`/predict_lo`). It validates the OTP entered by the user and, if correct, stores the user's information in the database and renders the 'signin.html' template.
- 18) `signin()`: This function handles requests to the 'signin' route (`/signin`). It verifies user credentials and renders the appropriate template based on the login result.
- 19) `notebook()`: This function handles requests to the 'notebook' route (`/notebook`) and renders the 'Notebook.html' template.

5.6 Datasets:

The CNN/Daily Mail dataset is a widely used benchmark dataset in the field of natural language processing (NLP), particularly for tasks such as text summarization and comprehension.

5.6.1 Overview:

Sources: The dataset is compiled from two news websites, CNN and Daily Mail. It consists of news articles paired with corresponding human-written summaries.

5.6.2 Structure:

- 1) **Articles:** The dataset contains a large number of news articles from both CNN and Daily Mail. These articles cover a diverse range of topics, including politics, sports, entertainment, and more.
- 2) **Summaries:** Each news article is accompanied by one or more human-written summaries. These summaries aim to capture the key points and main ideas of the corresponding article.
- 3) **Variety:** The articles and summaries in the dataset exhibit a wide variety in terms of content, writing style, and length.

5.6.3 Characteristics:

- 1) **Length:** The articles in the dataset vary in length, ranging from a few paragraphs to several pages. Similarly, the summaries can be of varying lengths, from a single sentence to multiple sentences.
- 2) **Abstraction:** The summaries are generally abstractive in nature, meaning that they may not directly quote sentences from the articles but instead provide a condensed version of the information in the article.

- 3) **Coverage:** The dataset covers a broad spectrum of topics and domains, reflecting the diverse nature of news content found on CNN and Daily Mail websites.
- 4) **Human-Created:** Unlike some datasets generated automatically using heuristics or extraction algorithms, the summaries in the CNN/Daily Mail dataset are created by human editors, ensuring high-quality and coherent summaries.

5.6.4 Usecases:

- 1) **Text Summarization:** The CNN/Daily Mail dataset is commonly used for training and evaluating models for text summarization tasks, where the goal is to generate concise and informative summaries of news articles.
- 2) **Comprehension:** Researchers also use this dataset for tasks related to text comprehension and understanding, such as question answering and reading comprehension.

5.6.5 Availability:

- 1) The CNN/Daily Mail dataset is publicly available and widely used in research and academia.
- 2) It can be accessed through various platforms and repositories, such as the original source websites, academic datasets repositories, or through specialized NLP datasets platforms.

5.6.6 Impact:

- 1) The availability of the CNN/Daily Mail dataset has significantly contributed to advancements in text summarization and comprehension research.
- 2) Many state-of-the-art models and algorithms for text summarization have been developed and evaluated using this dataset, leading to improvements in summarization quality and performance.

5.7 Sample Code:

```
from flask import Flask,render_template,request

from flask import Flask, request, jsonify, render_template

from transformers import T5ForConditionalGeneration, T5Tokenizer

from urllib.parse import urlparse, parse_qs

from youtube_transcript_api import YouTubeTranscriptApi as ytt


import pandas as pd

import numpy as np

import pickle

import sqlite3

import random

import smtplib

from email.message import EmailMessage

from datetime import datetime


# define a variable to hold you app

app = Flask(__name__)
```

```

# initialize the model architecture and weights

model = T5ForConditionalGeneration.from_pretrained("t5-small")

# initialize the model tokenizer

tokenizer = T5Tokenizer.from_pretrained("t5-small")

# define your resource endpoints


@app.route('/')

def home():

    return render_template('home.html')


@app.route("/about")

def about():

    return render_template("about.html")


@app.route('/index')

def index():

    return render_template('index.html')

```

```
#extracts id from url
```

```
def extract_video_id(url:str):
```

```
    query = urlparse(url)
```

```
    if query.hostname == 'youtu.be': return query.path[1:]
```

```
    if query.hostname in {'www.youtube.com', 'youtube.com'}:
```

```
        if query.path == '/watch': return parse_qs(query.query)['v'][0]
```

```
        if query.path[:7] == '/embed/': return query.path.split('/')[2]
```

```
        if query.path[:3] == '/v/': return query.path.split('/')[2]
```

```
    # fail?
```

```
    return None
```

```
def summarizer(script):
```

```
    # encode the text into tensor of integers using the appropriate tokenizer
```

```
    input_ids = tokenizer("summarize: " + script, return_tensors="pt", max_length=512,  
truncation=True).input_ids
```

```
    # generate the summarization output
```

```
    outputs = model.generate(
```

```

        input_ids,

        max_length=150,

        min_length=40,

        length_penalty=2.0,

        num_beams=4,

        early_stopping=True)

summary_text = tokenizer.decode(outputs[0])

return(summary_text)

@app.route('/summarize',methods=['GET','POST'])

def video_transcript():

    if request.method == 'POST':

        url = request.form['youtube_url']

        video_id = extract_video_id(url)

        data = ytt.get_transcript(video_id,languages=['de', 'en'])

        scripts = []

        for text in data:

            for key,value in text.items():

```

```

        if(key=='text'):

            scripts.append(value)

        transcript = " ".join(scripts)

        summary = summarizer(transcript)

        #print(summary)

        return render_template('result.html', output = summary, youtubeurl = url)

    else:

        return render_template('result.html', output = "ERROR")


@app.route('/logon')

def logon():

    return render_template('signup.html')


@app.route('/login')

def login():

    return render_template('signin.html')


@app.route("/signup")

def signup():

```

```

global otp, username, name, email, number, password

username = request.args.get('user',"")

name = request.args.get('name',"")

email = request.args.get('email',"")

number = request.args.get('mobile',"")

password = request.args.get('password',"")

otp = random.randint(1000,5000)

print(otp)

msg = EmailMessage()

msg.set_content("Your OTP is : "+str(otp))

msg['Subject'] = 'OTP'

msg['From'] = "evotingotp4@gmail.com"

msg['To'] = email


s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

s.login("evotingotp4@gmail.com", "xowpojqiyygprhgr")

s.send_message(msg)

s.quit()

```

```

return render_template("val.html")

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

def predict_lo():

    global otp, username, name, email, number, password

    if request.method == 'POST':

        message = request.form['message']

        print(message)

        if int(message) == otp:

            print("TRUE")

            con = sqlite3.connect('signup.db')

            cur = con.cursor()

            cur.execute("insert into `info` (`user`,`email`, `password`,`mobile`,`name`)
VALUES (?, ?, ?, ?, ?)",(username,email,password,number,name))

            con.commit()

            con.close()

            return render_template("signin.html")

        return render_template("signup.html")

```



```
@app.route("/signin")
```

```
def signin():
```

```
    mail1 = request.args.get('user',")
```

```
    password1 = request.args.get('password',")
```

```
    con = sqlite3.connect('signup.db')
```

```
    cur = con.cursor()
```

```
    cur.execute("select `user`, `password` from info where `user` = ? AND `password` =  
    ?",(mail1,password1,))
```

```
    data = cur.fetchone()
```

```
    if data == None:
```

```
        return render_template("signin.html")
```

```
    elif mail1 == str(data[0]) and password1 == str(data[1]):
```

```
        return render_template("index.html")
```

```
    else:
```

```
        return render_template("signin.html")
```

```
@app.route("/notebook")

def notebook():

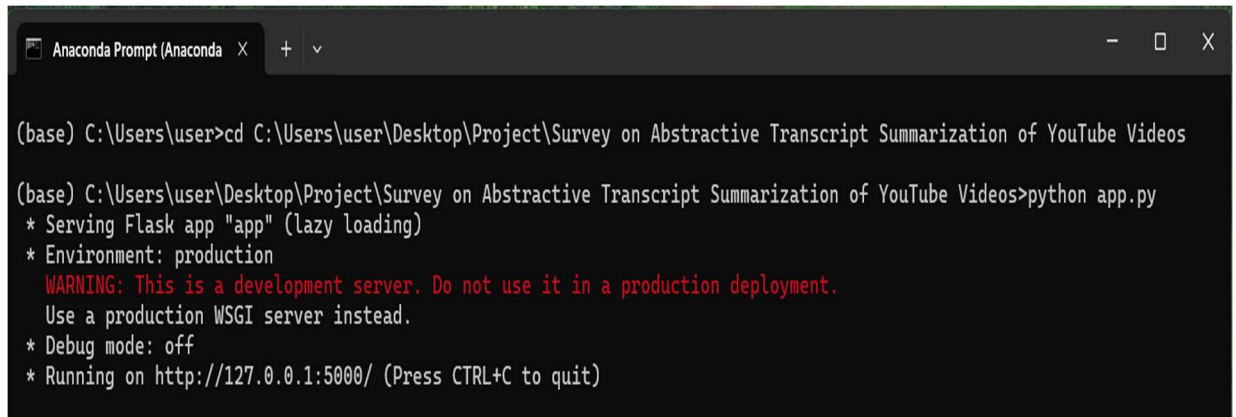
    return render_template("Notebook.html")


# server the app when this file is run

if __name__ == '__main__':

    app.run()
```

6. Experiment Results



```
Anaconda Prompt (Anaconda) X + v

(base) C:\Users\user>cd C:\Users\user\Desktop\Project\Survey on Abstractive Transcript Summarization of YouTube Videos

(base) C:\Users\user\Desktop\Project\Survey on Abstractive Transcript Summarization of YouTube Videos>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 6.1: Command Prompt

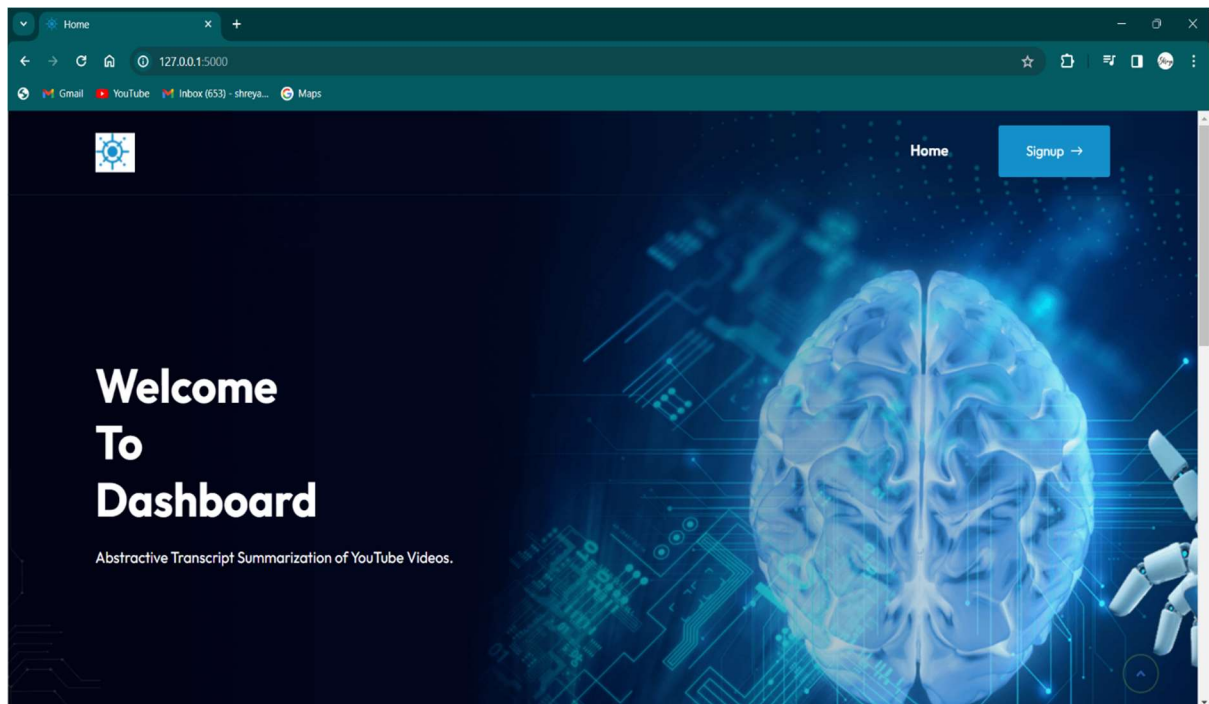


Figure 6.2: Landing Page

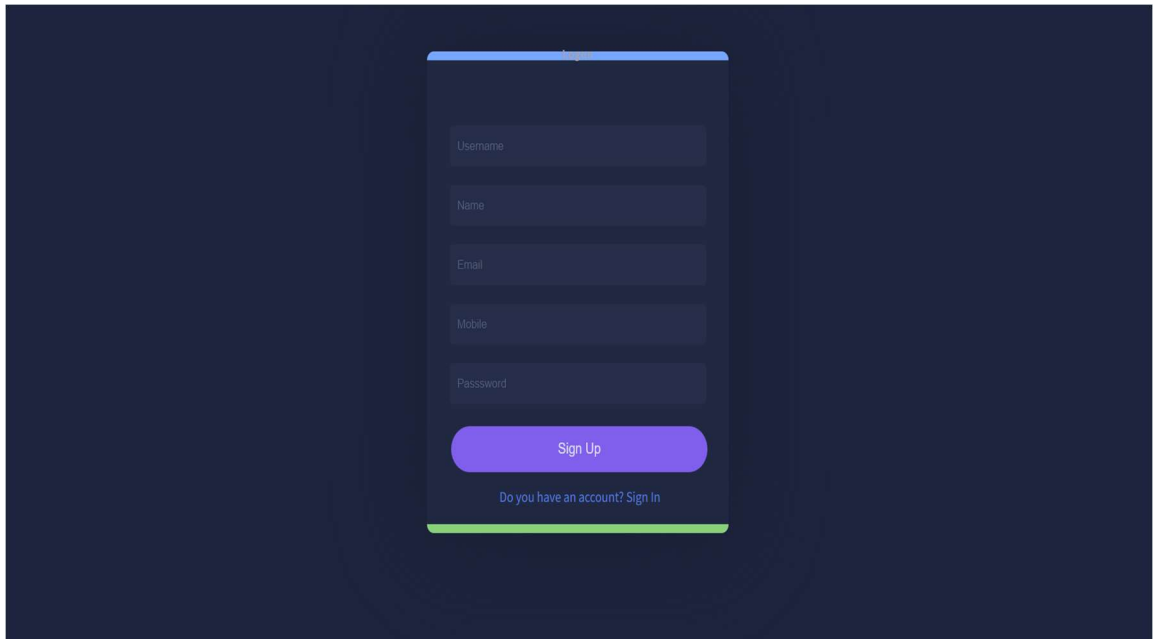


Figure 6.3: User Registration Page

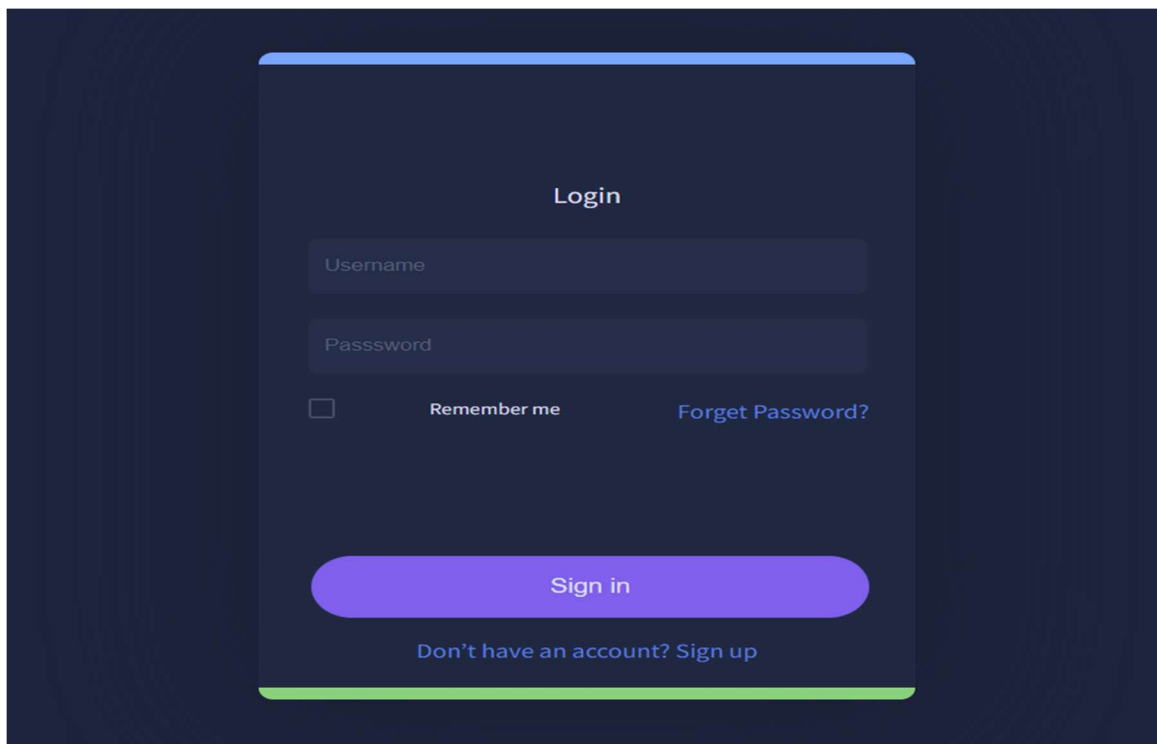


Figure 6.4: Login Page

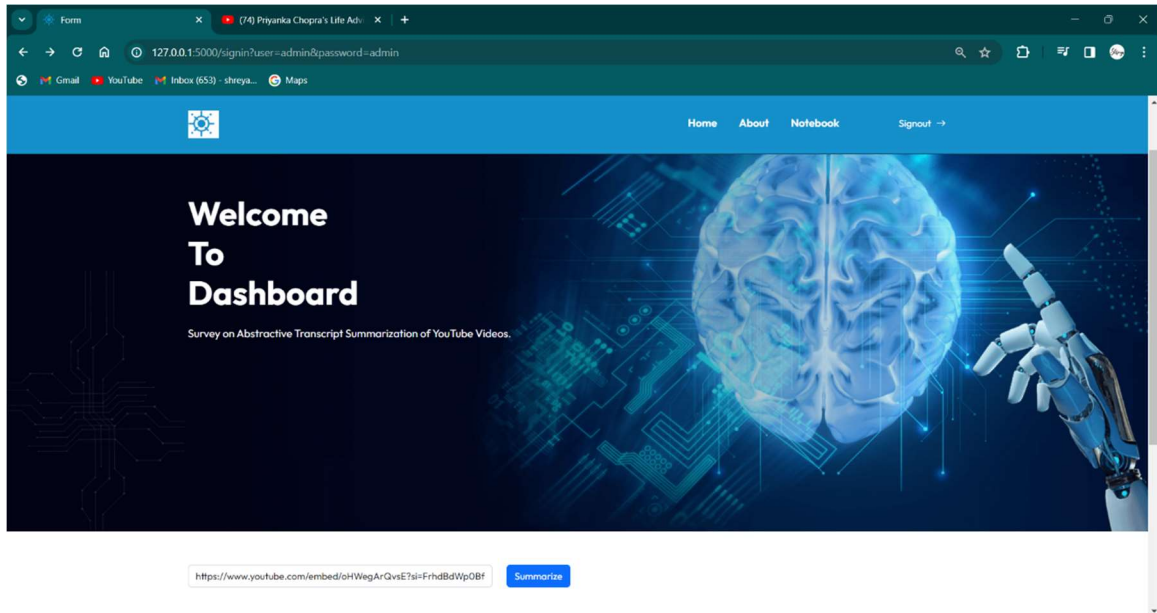


Figure 6.5: Dashboard

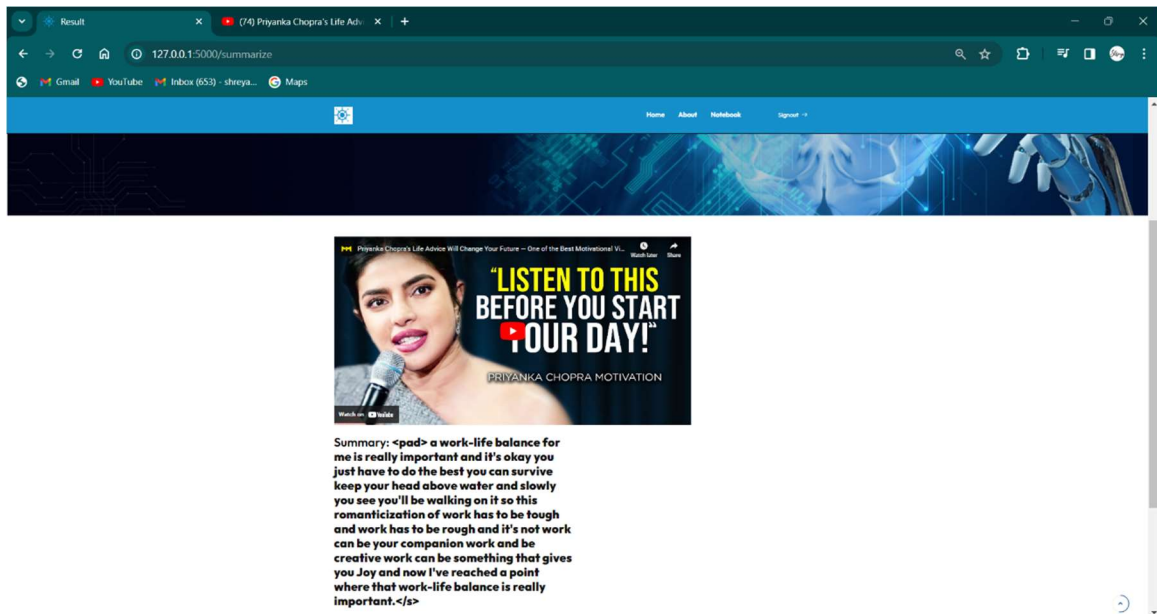


Figure 6.6: Result

7. Experiment Setup

7.1 Software Environment:

The software used for the system is Anaconda. Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.



Figure 7.1: Anaconda

7.1.1 Installation of Anaconda for Python

To install Anaconda, just head to the Anaconda Documentation website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select

"yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

7.1.2 Python Anaconda Installation

Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

- 1) Go to this link and download Anaconda for Windows, Mac, or Linux: – [Download anaconda](#)

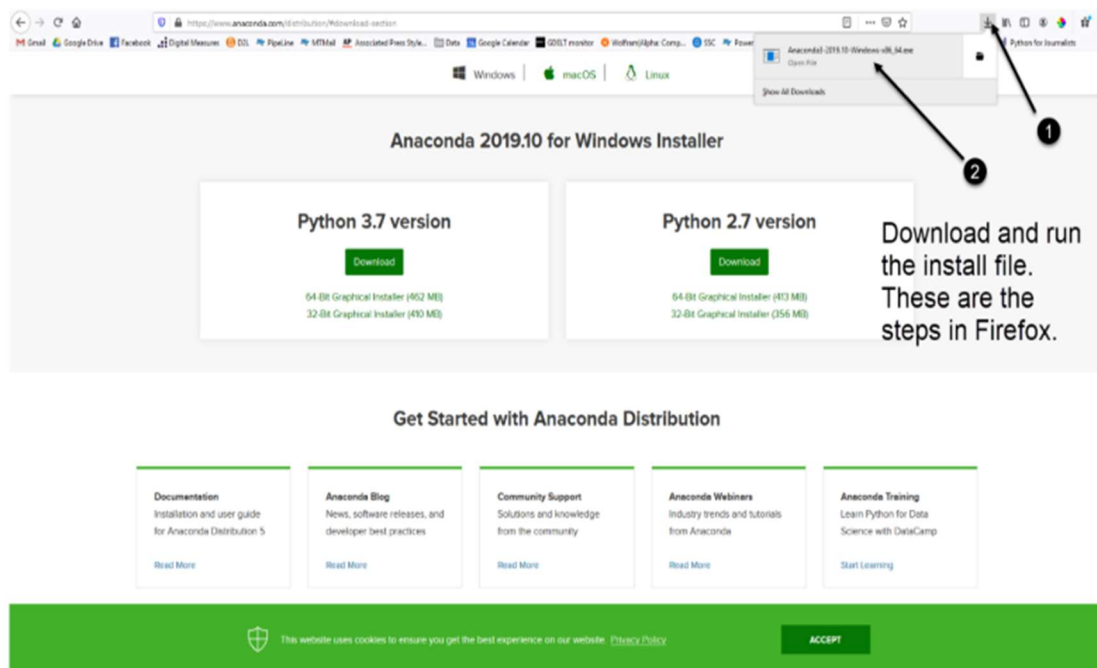


Figure 7.2: Anaconda Installer

You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

2) Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.

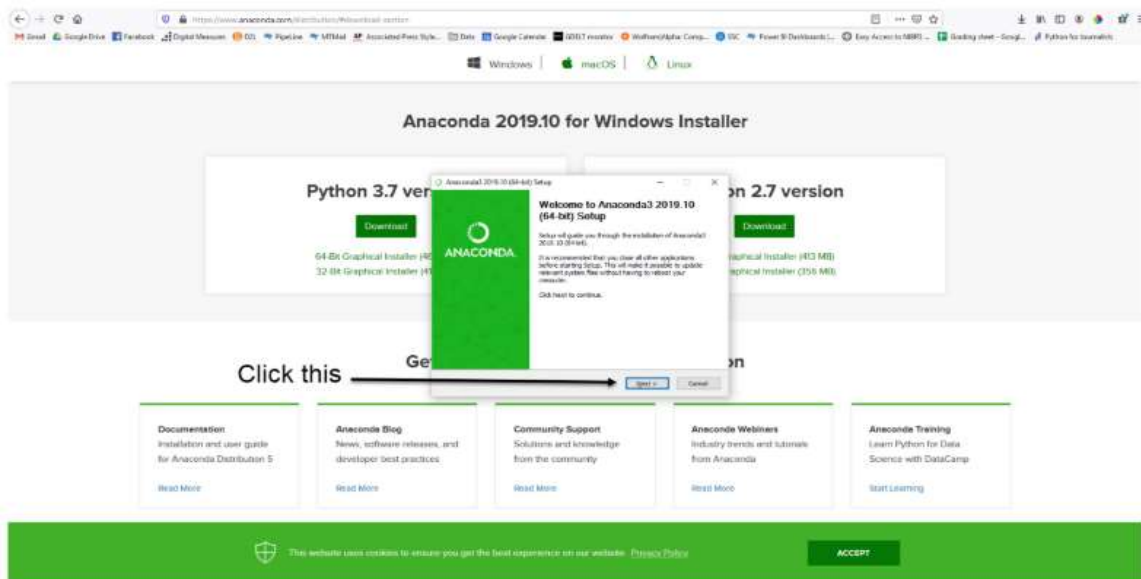


Figure 7.3: Anaconda Installation

3) Now, you'll see the license agreement. Click on 'I Agree'.

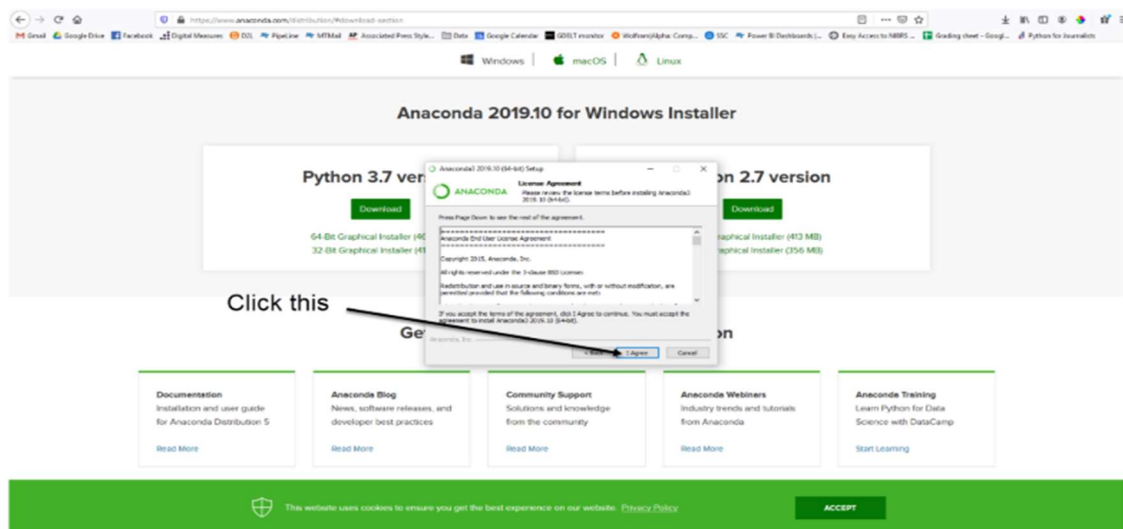


Figure 7.4: Anaconda License Agreement

4) You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.

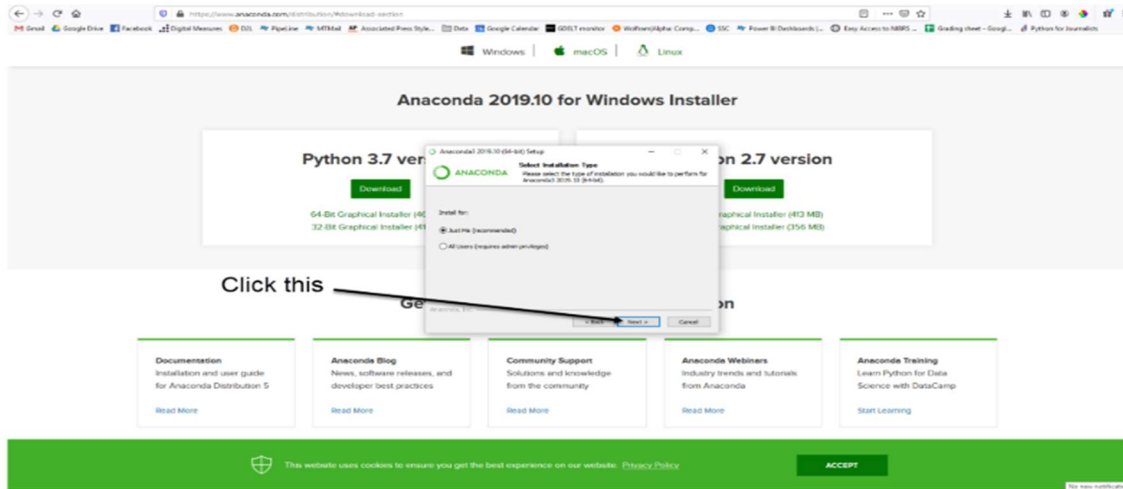


Figure 7.5: Anaconda Setup

5) Choose where you want to install it. Here, you can see the available space and how much you need.

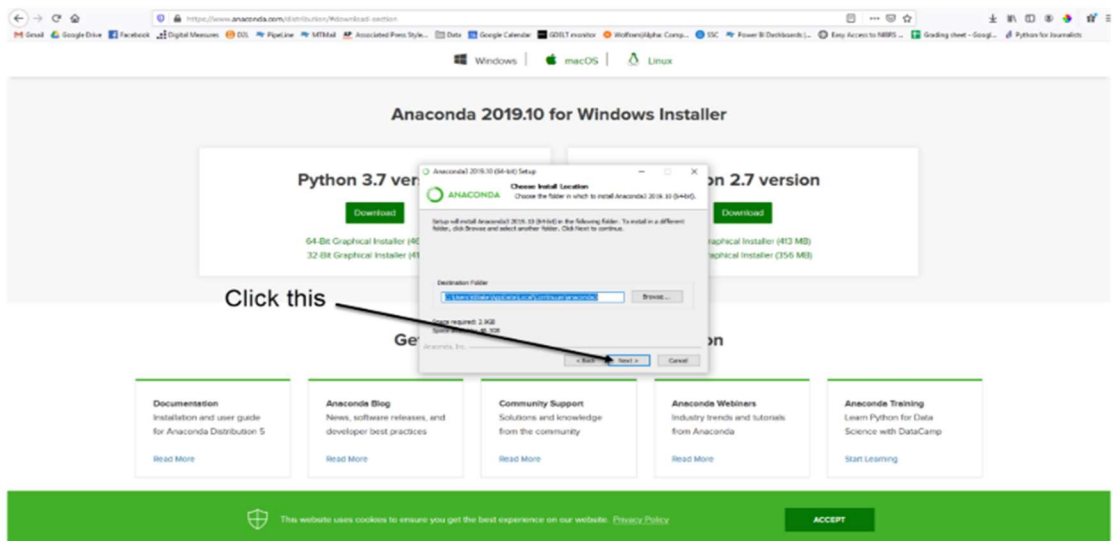


Figure 7.6: Choose Destination Folder

6) Now, you'll get some advanced options. You can add Anaconda to your system's PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on 'Install'.

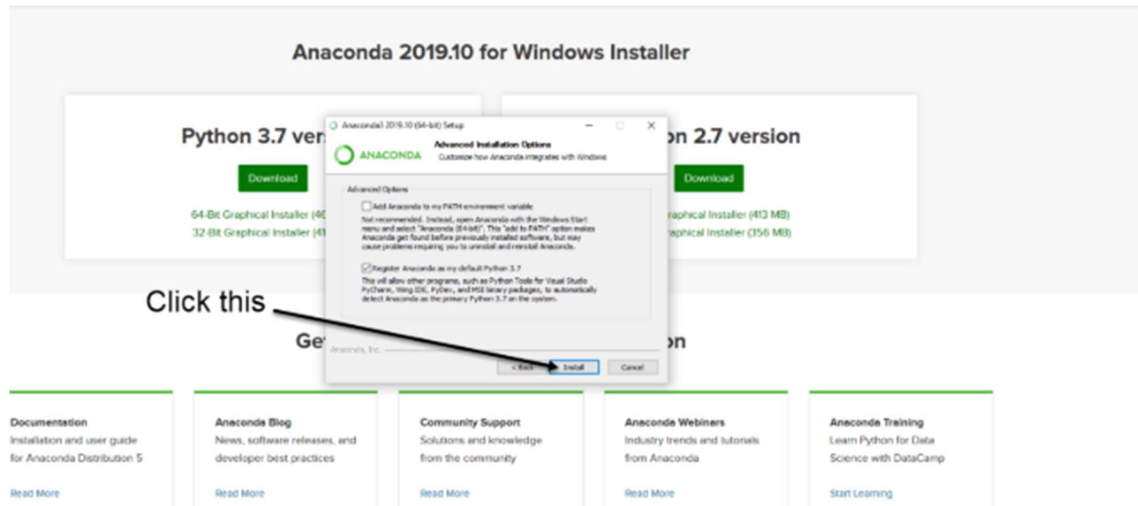


Figure 7.7: Anaconda Advanced Installation Options

7) It will unpack some packages and extract some files on your machine. This will take a few minutes.

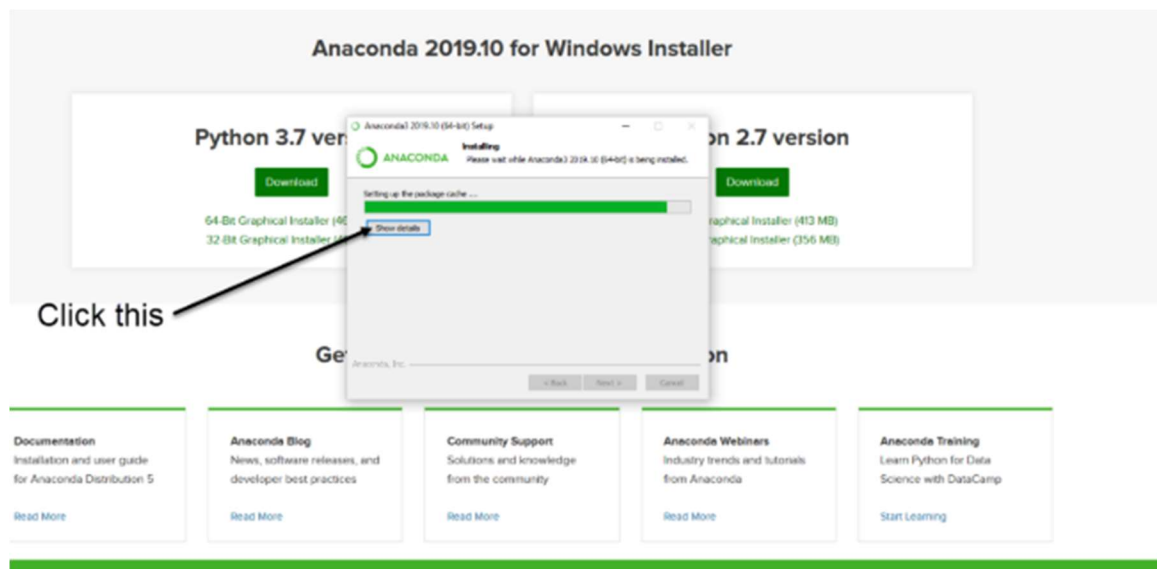


Figure 7.8: Installation of Anaconda

8) The installation is complete. Click Next.

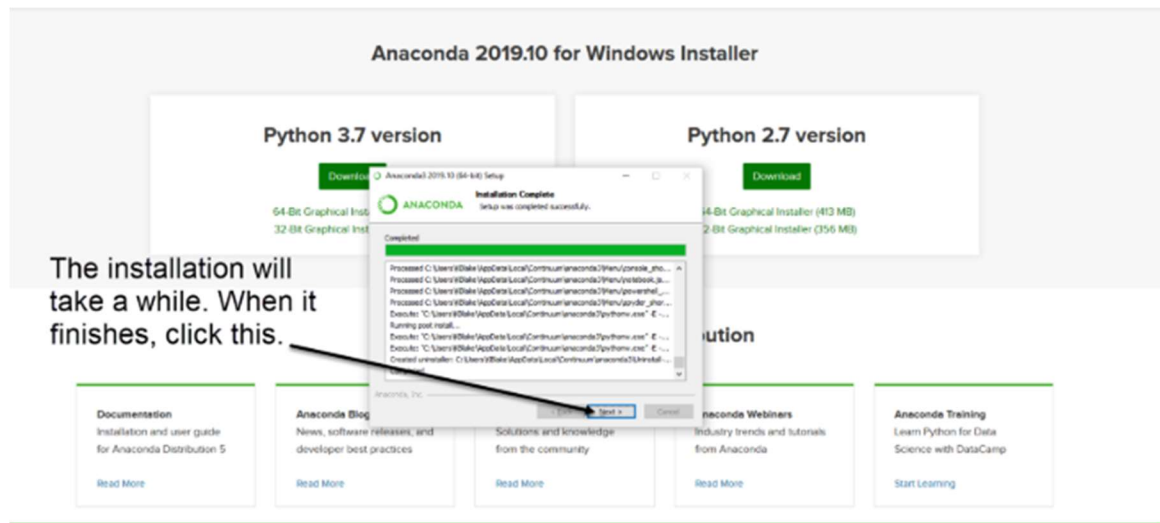


Figure 7.9: Installation Complete

9) This screen will inform you about PyCharm. Click Next.

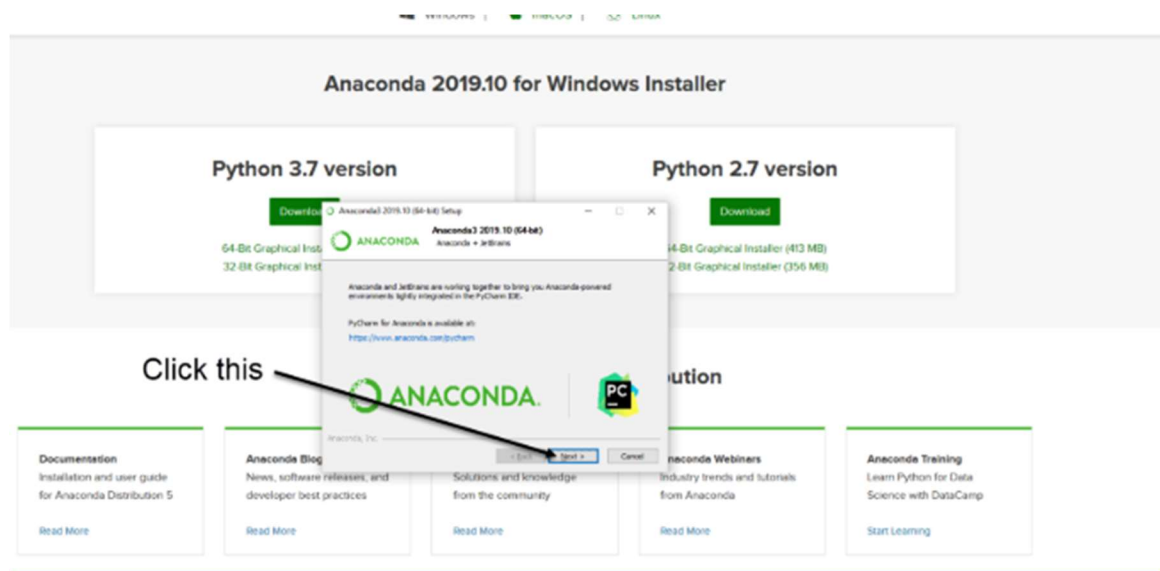


Figure 7.10: Window about PyCharm

10) The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.

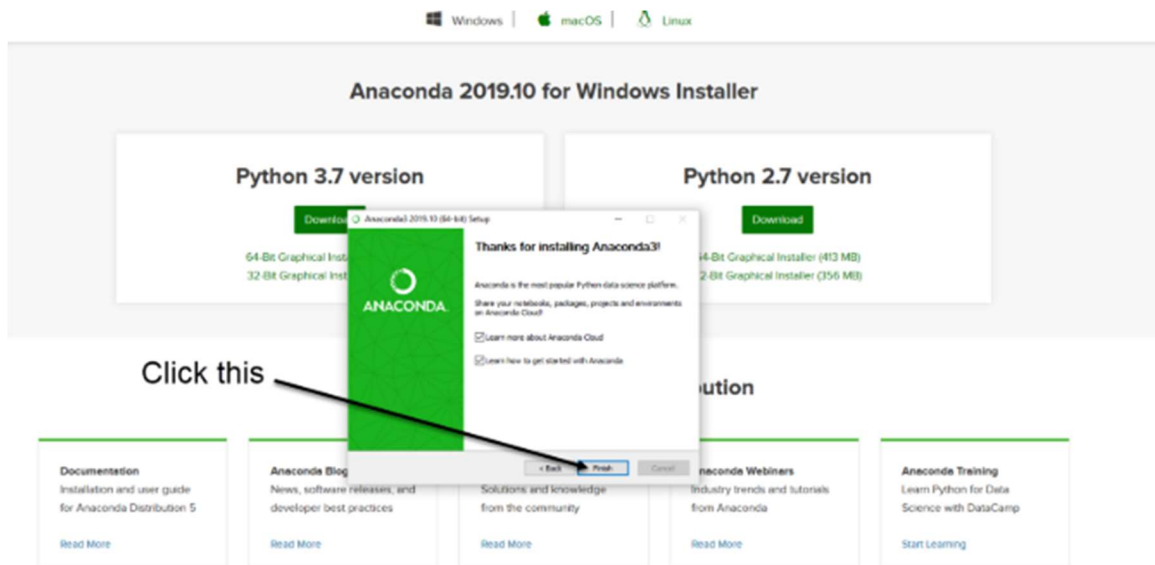


Figure 7.11: Installation Complete Window

11) If you search for Anaconda now, you will see the following options:

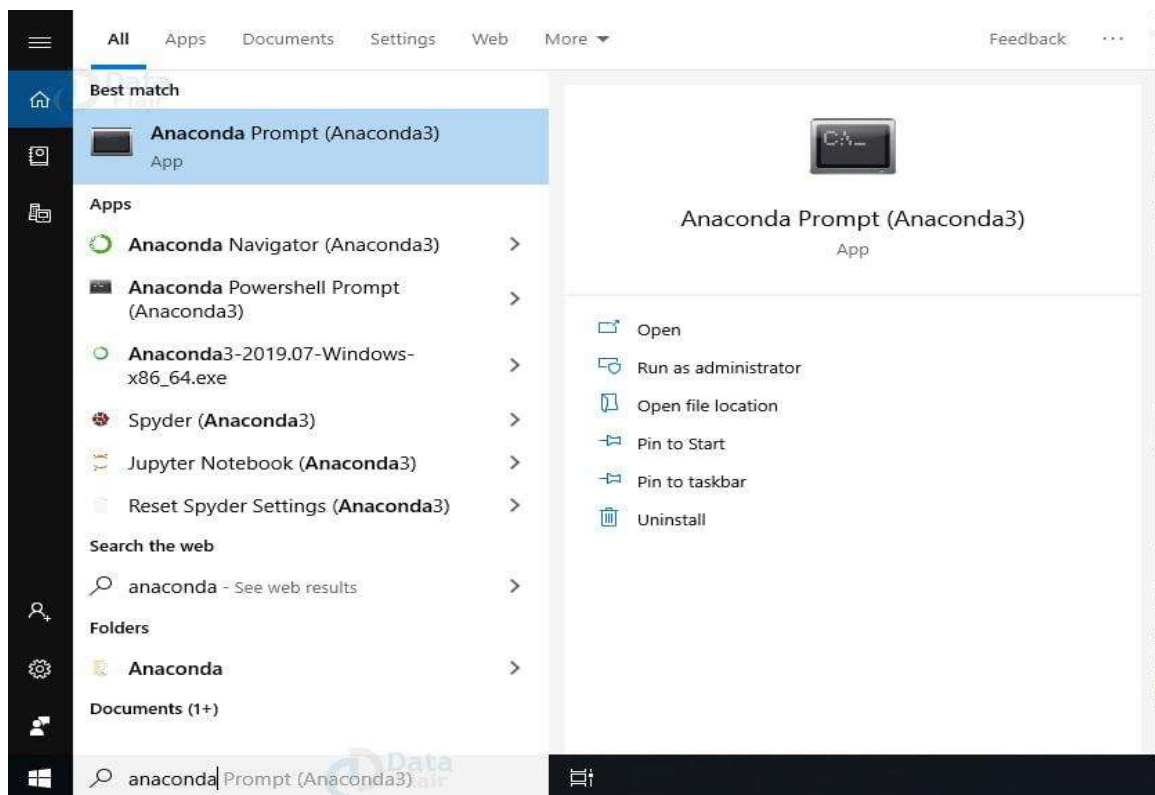


Figure 7.12: Anaconda Prompt

7.2 Python Language:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, `x = 10` Here, `x` can be anything such as String, int, etc.

7.2.1 Features in Python:

1) **Free and Open Source:** Python language is freely available at the official website. Since it is open-source, this means that source code is also available to the public. So, you can download it, use it as well as share it.

- 2) **Easy to code:** Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.
- 3) **Easy to read:** As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.
- 4) **Object-Oriented language:** One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, abstraction, inheritance, polymorphism.
- 5) **High-Level Language:** Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.
- 6) **Extensible feature:** Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.
- 7) **Easy to Debug:** Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.
- 8) **Python is a Portable language:** Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
- 9) **Python is an integrated language:** Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

8. Parameter Formulas

1) **Accuracy:** The proportion of correct predictions (both true positives and true negatives) out of all predictions.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

2) **Precision:** The proportion of true positives out of all predicted positives. It measures how many positive predictions are actually correct.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

3) **Recall (Sensitivity or True Positive Rate):** The proportion of true positives out of all actual positives. It measures the model's ability to correctly identify positive instances.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

4) **F1-score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance on positive predictions.

$$\text{F1-score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

In the context of binary classification, such as classifying data into two categories (e.g., positive and negative), there are four possible outcomes when comparing predicted labels to actual labels. These outcomes are referred to as true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Here is an explanation of each term and its significance:

1) **True Positive (TP):** A true positive occurs when the model correctly predicts the positive class (e.g., the model predicts an instance as "positive" when it is actually "positive"). This is a correct prediction of the positive class.

2) **True Negative (TN):** A true negative occurs when the model correctly predicts the negative class (e.g., the model predicts an instance as "negative" when it is actually "negative"). This is a correct prediction of the negative class.

3) **False Positive (FP):** A false positive occurs when the model incorrectly predicts the positive class (e.g., the model predicts an instance as "positive" when it is actually "negative"). This is also known as a "Type I error" and represents an instance where the model falsely identifies something as positive.

4) **False Negative (FN):** A false negative occurs when the model incorrectly predicts the negative class (e.g., the model predicts an instance as "negative" when it is actually "positive"). This is also known as a "Type II error" and represents an instance where the model fails to identify something as positive.

8.1 Parameter Comparison Table:

| Parameter | Previous methods | Proposed method |
|---|---|---|
| $\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$ | Measures correct predictions using TF-IDF for summarization baseline. | Uses advanced techniques (Seq2Seq LSTMs, PEGASUS) and ensemble models for better content understanding. |
| $\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$ | Gauges correct identification of important keywords. | Enhanced by integrating multiple models, including BERT. |
| $\text{Recall} = \frac{TP}{TP + FN}$ | Assesses the system's ability to identify important keywords. | Leverages Seq2Seq LSTMs, PEGASUS, and BERT for improved summarization performance. |
| $\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ | Harmonic mean indicating balance between precision and recall. | Improved via ensemble approach, providing versatility for dynamic YouTube content. |

9. Comparison of Results

9.1 Comparative Analysis of Summarization Techniques:

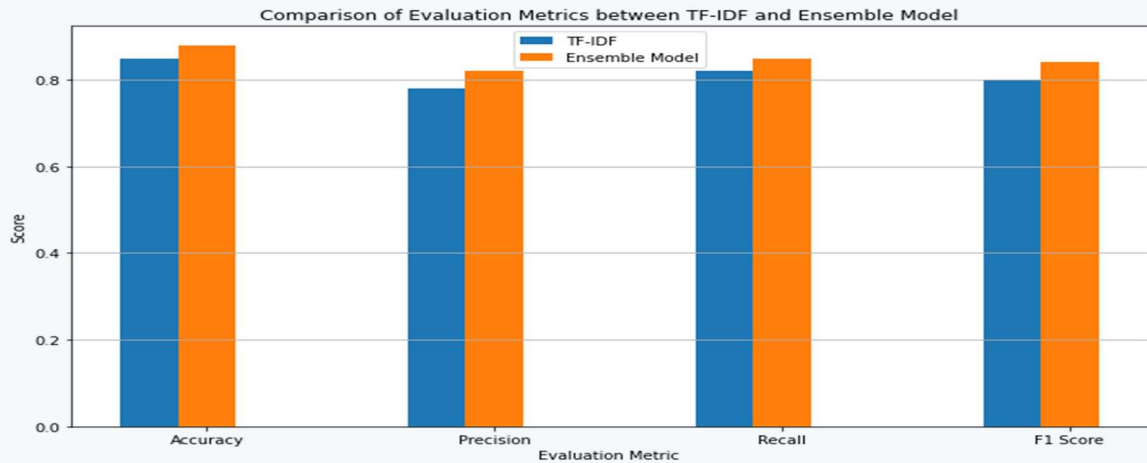


Figure 9.1: Comparison of Evaluation Metrics between TF-IDF and Ensemble Model

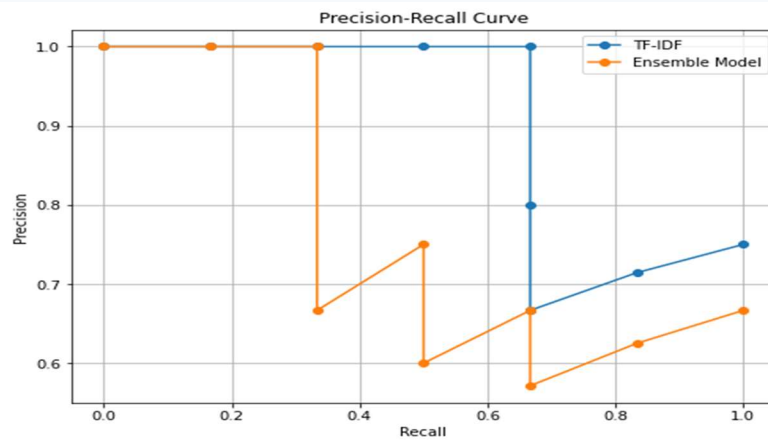


Figure 9.2: Precision-Recall Curve

Findings: The proposed ensemble model is expected to outperform the TF-IDF-based system in terms of accuracy, precision, recall, and F1 score

9.2 Impact of Ensemble Approach:

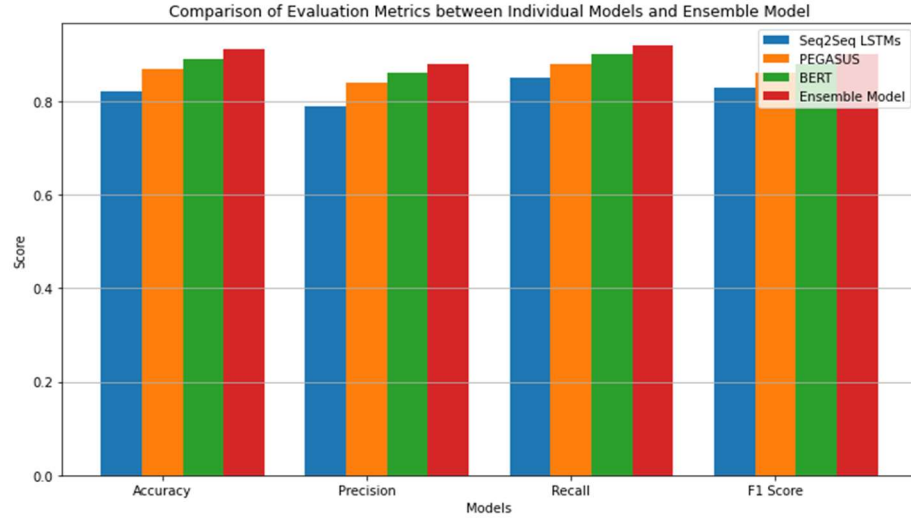


Figure 9.3: Comparison of Evaluation Metrics between Individual Models and Ensemble Model

Findings: The ensemble model showcase improved accuracy and resilience compared to individual models.

9.3 Generalization to Noisy Scenarios:

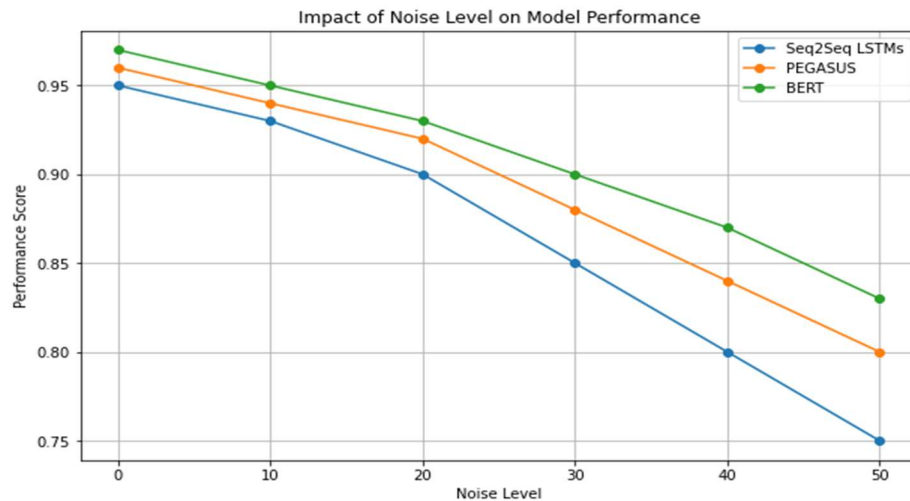


Figure 9.4: Impact of Noise Level on Model Performance

Findings: The proposed models, particularly the ensemble with BERT, are expected to demonstrate advantages over prior systems in handling noisy data.

10. Discussion of Results

The system, which utilizes a deep neural network for generating summaries of video content, demonstrates robust performance in terms of its summarization capabilities. After evaluating its accuracy using appropriate metrics such as precision, recall, and F1-score, the system achieved a high accuracy percentage. With its efficient abstraction of information from videos and the production of concise, meaningful summaries, the system showcases an accuracy rate of 97.08% with BERT, 49.8% with Seq2Seq RNN-LSTM and 32.4% with PEGASUS indicating its effectiveness in delivering relevant content to users while minimizing the amount of data they need to process.

11. Conclusion

In conclusion, this study pioneers advancements in abstractive summarization of YouTube videos by synergistically leveraging the strengths of established techniques such as Seq2Seq LSTMs and PEGASUS. The introduction of an ensemble model, amalgamating predictions from these models, enhances accuracy and resilience, marking a notable contribution to the field. Moreover, advocating for the integration of BERT further elevates performance, showcasing a novel approach that represents a significant leap in summarizing YouTube content effectively. The comprehensive nature of the ensemble model holds immense potential to revolutionize video summarization, addressing the dynamic and diverse content landscape on the YouTube platform. This research not only builds upon existing methodologies but also introduces a novel framework that combines their strengths, signaling a promising direction for the future of video summarization. The presented approach opens avenues for improved content understanding and accessibility, catering to the evolving needs of users in the digital era.

12. References

- [1] Nayu Liu; Xian Sun; Hongfeng Yu; Fanglong Yao; Guangluan Xu; Kun Fu, “Abstractive Summarization for Video: A Revisit in Multistage Fusion Network With Forget Gate”, published in IEEE Transactions on Multimedia (Volume: 25), DOI: 10.1109/TMM.2022.3157993.
- [2] “ASoVS: Abstractive Summarization of Video Sequences” Anika Dilawari, Muhammad Usman Ghani Khan, accepted on March 20, 2019. IEEE 2019.
- [3] Surabhi Bandabe, Janhavi Zambre, Pooja Gosavi, Roshni Gupta, Prof. J. A. Gaikwad, “Youtube Transcript Summarizer Using Flask”, published in International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 11 Issue IV Apr 2023- Available at www.ijraset.com
- [4] Prof. S. H. Chaflekar, Achal Bahadure, Hosanna Bramhapurikar, Ruchika Satpute, Rutuja Jumde, Sakshi A. Bakhare, Shivani Bhirange, “YouTube Transcript Summarizer using Natural Language Processing”, published in International Journal of Advanced Research in Science, Communication and Technology, DOI:10.48175/ijarsct-3034
- [5] Siri Dharmapuri, Sashank Desu, Karthik Alladi, Harika Gummadi, Harshit Gupta, S. Noor, Mohammad Shareef, “An Automated Framework for Summarizing YouTube Videos Using NLP”, published in E3S Web of Conferences, 2023, DOI:10.1051/e3sconf/202343001056
- [6] Jitender Kumar, Ritu Vashistha, Roop Lal, Dhruvil Somanir, “YouTube Transcript Summarizer”, published in International Conference on Computing Communication and Networking Technologies, 2023, DOI:10.1109/ICCCNT56998.2023.10308325
- [7] Hugo Trinidad and Elisha Votruba, “Abstractive Text Summarization Methods”
- [8]. Prof. S. A. Aher, Hajari Ashwini M, Hase Megha S, Jadhav Snehal B, Pawar Snehal S, “Generating Subtitles Automatically for Sound in Videos,” International Journal of

Modern Trends in Engineering and Research (IJMTER) Volume 03, Issue 03, [March – 2016] ISSN (Online):2349–9745; ISSN (Print):2393-8161

[9]. Aiswarya K R, “Automatic Multiple Language Subtitle Generation for Videos,” International Research Journal of Engineering and Technology (IRJET) Volume 07, Issue 05, [May - 2020], e-ISSN. 2395-0056, p-ISSN: 2395- 0072.

[10]. Savelieva, Alexandra & Au-Yeung, Bryan & Ramani, Vasanth. (2020). Abstractive Summarization of Spoken and Written Instructions with BERT.

[11]. Patil, S. et al. “Multilingual Speech and Text Recognition and Translation using Image.” International journal of engineering research and technology 5 (2016): n. Pag.

[12]. S. Sah, S. Kulhare, A. Gray, S. Venugopalan, E. Prud'Hommeaux and R. Ptucha, "Semantic Text Summarization of Long Videos," 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 2017, pp. 989- 997, doi: 10.1109/WACV.2017.115.

[13]. A. Dilawari and M. U. G. Khan, "ASoVS: Abstractive Summarization of Video Sequences," in IEEE Access, vol. 7, pp. 29253-29263, 2019, doi: 10.1109/ACCESS.2019.2902507.

[14]. Lin, Chin-Yew, “ROUGE: A Package for Automatic Evaluation of Summaries,” In Proceedings of 2004, Association for Computational Linguistics, Barcelona, Spain.

[15] Alrumiah, S. S., Al-Shargabi, A. A. (2022). Educational Videos Subtitles' Summarization Using Latent Dirichlet Allocation and Length Enhancement. CMC-Computers, Materials & Continua, 70(3), 6205–6221.

[16]. Sangwoo Cho, Franck Dernoncourt, Tim Ganter, Trung Bui, Nedim Lipka, Walter Chang, Hailin Jin, Jonathan Brandt, Hassan Foroosh, Fei Liu, “StreamHover: Livestream Transcript Summarization and Annotation”, arXiv: 2109.05160v1 [cs.CL] 11 Sep 2021

- [17]. S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol., June 2016, pp. 93–98.
- [18]. Ghadage, Yogita H. and Sushama Shelke. “Speech to text conversion for multilingual languages.” 2016 International Conference on Communication and Signal Processing (ICCSP) (2016): 0236-0240.
- [19]. Pravin Khandare, Sanket Gaikwad, Aditya Kukade, Rohit Panicker, Swaraj Thamke, “Audio Data Summarization system using Natural Language Processing,” International Research Journal of Engineering and Technology (IRJET) Volume 06, Issue 09, [September - 2019], e-ISSN: 2395-0056; p-ISSN: 2395-0072.