GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

AUTOMATIC TEXT SUMMARIZATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES.

A Project Report Submitted In Partial Fulfilment Of The

Requirement For The Award Of Degree Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

Submitted by

MD. FASI AHAMED	**19341A1273**
M.V RAM KUMAR	**19341A1277**
S. MOHAN	**20345A1206**

Under the esteemed guidance of

Dr. Ajit Kumar Rout

Professor & HOD, Dept. of Information Technology

NOVEMBER 2022

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2015 Certified Institution)

GMR Nagar, Rajam - 532 127,

Andhra Pradesh, India

2022-2023

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

Department of Information Technology

CERTIFICATE

This is to certify that the thesis entitled **AUTOMATIC TEXT SUMMARIZATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES.** Submitted by **MD. FASI AHAMED (19341A1273), M.V. RAM KUMAR (19341A1277), S. MOHAN (20345A1206),** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **Information Technology** of **GMRIT, Rajam** affiliated to **JNTUK, KAKINADA** is arecord of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for theaward of any degree.

Signature of Supervisor
Dr. Ajit Kumar Rout
Professor & Head
Department of IT
GMRIT, Rajam.

Signature of HOD
Dr. Ajit Kumar Rout
Professor & Head
Department of IT
GMRIT, Rajam.

The report is submitted for the viva-voce examination held on ……..

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to our guide **Dr. Ajit Kumar Rout,** Professor & Hod, Department of Information Technology of whole hearted and invaluable guidance throughout the project work. Without his sustained and sincere effort, this project work would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. Ajit Kumar Rout**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr.C.L.V.R.S.V. Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank the project coordinator **Mr. Ch. Anil Kumar**, all the faculty members and the non-teaching staff of the Department of Information Technology for theirdirect or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

MD FASI AHAMED (19341A1273)
M.V RAM KUMAR (19341A1277)
S MOHAN (20345A1206)

ABSTRACT

Text Summarization has always been a great area of research in the Artificial Intelligence. Text summarization is the process of making a synopsis from a given text document while keeping the important information and meaning of it. Automatic summarization has become an essential method for accurately locating significant information in vast amounts of text in a short amount of time and with minimum effort. In this project, we propose a different Text Summarization model based on NLP, focusing on the SPACY & NLTK techniques and analyzing their influence on the sentence generation. SPACY is used to build information extraction or natural language understanding systems. NLTK is an essential library supports tasks such as classification, stemming, semantic reasoning, and tokenization in Python. Moreover applying Text Summarization gears up the procedure of researching, reduces the reading time, and increases the amount of important information. Finally the output of this project is a web application that can summarize a text or a Wikipedia link. We have additionally been given an opportunity to compare different methods of summarization.

Keywords: *Natural Language Processing (NLP), Summary, spaCy, Natural Language Toolkit (NLTK).*

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NLP	:	Natural Language Processing
OOP	:	Object Oriented Programming
HTML	:	Hypertext Markup Language
CSS	:	Cascading Style Sheets
NLTK	:	Natural Language Tool Kit
URL	:	Uniform Resource Locator
SQL	:	Structured Query Language
ELM-AE	:	Extreme Learning Auto Encoder
LSTM	:	Long Short Term Memory
GRU	:	Gated Recurrent Unit
TIF-SR	:	Topic Information Fusion and Semantic Relevance
API	:	Application Programming Interface
QA	:	Question Answering
FAQs	:	Frequently Asked Questions
MDE	:	Model Driven Engineering
TF-IDF	:	Term Frequency Inverse Document Frequency
GloVe	:	Global Vectors for Word Representation
seq2seq	:	Sequence to Sequence
POS	:	Parts of Speech

Chapter 1

INTRODUCTION

1.1 Natural Language Processing

NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence. It is the technology that is used by machines to understand, analyze, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation. NLP enables computers to understand natural language as humans do. Whether the language is spoken or written, natural language processing uses artificial intelligence to take real-world input, process it, and make sense of it in a way a computer can understand. Just as humans have different sensors -- such as ears to hear and eyes to see -- computers have programs to read and microphones to collect audio. And just as humans have a brain to process that input, computers have a program to process their respective inputs. At some point in processing, the input is converted to code that the computer can understand.

Now, modern NLP consists of various applications, like speech recognition, machine translation, and machine text reading. When we combine all these applications then it allows the artificial intelligence to gain knowledge of the world. Let's consider the example of AMAZON ALEXA, using this robot you can ask the question to Alexa, and it will reply to you.

1.2 Text Summarization

Text summarization is a very useful and important part of Natural Language Processing (NLP). First let us talk about what text summarization is. Suppose we have too many lines of text data in any form, such as from articles or magazines or on social media. We have time scarcity so we want only a nutshell report of that text. We can summarize our text in a few lines by removing unimportant text and converting the same text into smaller semantic text form. Actually abstractive summarization is complex task when compared to extractive approach, In this approach we build algorithms or programs which will reduce the text size and create a summary of our text data. This is called automatic text summarization in machine learning. Text summarization is the process of creating shorter text without removing the semantic structure of text.

There are two approaches to text summarization.

1.2.1 Extractive approaches:

Using an extractive approach we summarize our text on the basis of simple and traditional algorithms. For example, when we want to summarize our text on the basis of the frequency method, we store all the important words and frequency of all those words in the dictionary. On the basis of high frequency words, we store the sentences containing that word in our final summary. This means the words which are in our summary confirm that they are part of the given text.

1.2.2 Abstractive approaches:

An abstractive approach is more advanced. On the basis of time requirements we exchange some sentences for smaller sentences with the same semantic approaches of our text data. When abstraction is used for text summarization in deep learning issues, it can overcome the extractive method's grammatical errors. Abstraction is more efficient than extraction. The text summarizing algorithms necessary for abstraction, on the other hand, are more complex to build, which is why extraction is still widely used.

1.3 Twitter

Twitter is a microblogging and social networking service owned by American company Twitter, Inc., on which users post and interact with messages known as "tweets". Registered users can post, like, and retweet tweets, while unregistered users only have a limited ability to read public tweets. Users interact with Twitter through browser or mobile frontend software, or programmatically via its APIs. Prior to April 2020, services were accessible via SMS. Tweets were originally restricted to 140 characters, but the limit was doubled to 280 for non-CJK languages in November 2017. Audio and video tweets remain limited to 140 seconds for most accounts. Twitter's primary purpose is to connect its users and allow them to share their thoughts with their followers and others through the use of hashtags. It can be a source of news, entertainment and a marketing tool for businesses.

1.3.1 Tweet: A tweet is a post on Twitter. The act of writing a tweet is called tweeting or twittering. Tweets can be up to 140 characters long, including spaces, and can include URLs and hashtags. The Tweet consists a chirping note that is to make a posting on the Twitter online message service. A Tweet containing another account's Twitter username, preceded by the "@" symbol. For example: "Hello @TwitterSupport!"

1.3.2 Hashtag (#): On Twitter, adding a "#" to the beginning of an unbroken word or phrase creates a hashtag. When you use a hashtag in a Tweet, it becomes linked to all of the other Tweets that include it. Including a hashtag gives your Tweet context and allows people to easily follow topics that they're interested in.

1.3.3 Twitter Developer Platform: Twitter's Developer Platform enables you to harness the power of Twitter's open, global, real-time and historical platform within your own applications. The platform provides tools, resources, data and API products for you to integrate, and expand Twitter's impact through research, solutions and more.

1.3.4 Twitter API: The Twitter API is a set of programmatic endpoints that can be used to understand or build the conversation on Twitter. The Twitter API v2 includes a few access levels to help you scale your usage on the platform. This API allows you to find and retrieve, engage with, or create a variety of different resources including the Tweets, Users, Spaces, Direct Messages, Lists, Trends, Media, and Places.

1.3.5 Authentication Keys: Twitter APIs handle enormous amounts of data. The way we ensure this data is secured for developers and users alike is through authentication. There are a few methods for authentication, such as OAuth 1.0a User Context, App only, Basic authentication and OAuth 2.0 Authorization Code Flow with PKCE. Your App's API Keys and Bearer Token, as well as your personal Access Token and Access Token Secret can be obtained from the Twitter developer Apps section found in the developer portal.

1.4 Web Application Development

From the inception of modern programming languages in the early 50s of the 20th century, numerous programming languages have been invented. A few have been discontinued along the road, and a few have survived with appreciation from computer

scientists, programmers, and engineers. Of those survived, languages such as C, C++, Java, Python, JavaScript, and Ruby have been the go-to programming languages to build small to enterprises applications for desktop, mobile and other handheld devices.

While building a software application, one of the most important tasks of a programmer is to make the code readable, understandable and reusable. Often termed as DRY methodology, adopting this methodology not only reduces the boilerplate code in the code-base but also organizes the same logic in one place and makes it reusable in other parts of the codebase for the same purpose.

Another difficulty for the programmer is to understand the code which is written by other fellow programmers. If the same logic is repeated and found in many places, following the small piece of code could be a very demanding task as the source of truthfor the same can be found in many places in the codebase resulting in multiple interpretations increasing the complexity. Using the software frameworks, boilerplate, duplicate and un-standardized code can be reduced.

A software framework is a set of standardized libraries and tools for a programming language. It helps to build a concise software application efficiently. Because of its code reusability and efficiency, it allows a programmer to bootstrap an application in no time and start implementing the business logic. Software frameworks come in different types for different programming languages. In this thesis, however, we will try to compare and uncover the various aspects of the two most popular web application frameworks of Python programming language, i.e. Flask and Django. According to the documentation, Flask is as a micro framework. Because of it being lightweight and having the features of flexibility and extensibility, it can be bootstrapped in no time. On the other hand, Django is known as a "battery included" framework, which claims to have most of the required extensions and libraries to boot- strap a generic application providing a developer more time on implementing the business logic.

1.4.1 Background

Web application development is a process of developing software applications that can

run on websites. Even though web application development follows the software development process, the technology and the architecture used for it is quite different. The software application that runs on a personal computer (PC) might not depend on the internet in contrast to the web application that depends upon the remoteservers.

The web applications and the client-server technology have come quite far in comparison to the simple standalone phone book app made by Tim Berners-Lee in 1989. Now a-days, web application comes in different shapes and sizes such as static, dynamic, con-tent management system, e-commerce, and gaming to live content sharing portals. Commonly shared technology of applications mentioned above types is the backend and frontend technology.

1.5 Web Development Structure

The entire development process has been subdivided into two: the front end development and the backend development. The front end comprises of the visually visible parts such as the home page, contact page, admin panel, shopping cart page. The back end contains the database and its interaction with the front-end.

1.5.1 Front End Development

The front end was initially raw coded using JavaScript. JavaScript is a client side scripting language which is a dedicated language for web development. JavaScript code was simply mixed with the Hypertext Mark-up Language (HTML5) code. A static page is an HTML5 document that is stored on the web server and does not change. Hypertext mark-up language is the language used to design the web pages of an application. Cascading Style Sheet (CSS) is a style sheet language used for describing the look andformatting a document written in a mark-up language. These CSS files are linked with the class files with php extensions to put the panels in order, the text with correct font,size and color. JavaScript is a client side scripting language most commonly used as part of web browsers and its implementations allow client side scripts to interact with the user, control the browser and alter the document content which is displayed. For example, in website for the client's registration, the system asks to provide their detailswhich contains their name, email address, mobile number, etc. If they missed any of the details, then immediately the browser asks them to fill the particular field. This is implemented & handled by a JavaScript.

On the other hand, frontend development is mostly concerned with the aesthetic and the content displaying part, which is also known as client-side development. One of its difficult challenges is to be able to show the material in the different types of devices and browser. Many devices have its own Software Development Kit (SDK) which should be followed to serve the same content that is served in the browser. Website design, usability and user-friendliness are the essential factors that are addressed during the development. The figure below is an example of how the frontendside of an application works.

1.5.2 Backend Development

Backend development deals with the logical side of the web application. It mainly concerns the programming languages, core architecture and the logics. Those logics are mainly written in programming languages that can run on computer servers. It alsoinfluences how the data is stored, accessed and served from the servers. the backend part of the application, which consists of server-side scripts, frameworks, database andAPIs.

The Database Management System (DBMS) provides support for the back end. The database management system is essentially software where admin can create the database, add, drop, alter and update tables. The tables can hold different types of data for example: integer, variable characters etc. in our application we have chosen the MySQL DBMS to hold the database. MySQL is a relational database management system. The main reason is MySQL development project has made its source code available under the terms of the General Public License (GNU) which is an open sourceweb application.

1.6 Development Tools

1.6.1 HTML

HTML is HyperText markup language. It is an emerging technology, cascading style sheets, could eliminate many of the HTML table could be used to control the layout of a webpage. A web designer might separate the header, body text, and sidebarof a webpage by putting each into a distinct cell. Additionally, the net designer could put each link button on the header and sidebar into a separate cell so he or she could define unique properties for every button. Then, within the body of the page, the net designer could separate the textual

and graphical elements into different cells to regulate spacing and other attributes individually.

HTML has wide ranges of support for the contents from text to spreadsheets, video clips, and animations pictures. The web page often called as a document in a technical term, the document type defined at the beginning of the page determines what type of document it is and how it should be taken by the web browser while rendering it. Being the essential part of the World Wide Web, it has gone through the various phases of development from 1991 till date, while HTML being the initial version and HTML5 being the latest one respectively.

1.6.2 CSS3

CSS may be a formatting language want to add styling to your page. This can bedone by having the CSS document linked into your html page. This page then has selectors and properties which affect the tags inside your html document.CSS was introduced in 1996. It had been created to prevent people from having to repeat plenty of code. For instance, if someone wanted to alter the paragraph text, they'd should haveintercourse every single time they wanted to alter the properties. CSS has since become more adapted to having more features, for instance we will now use the tools and alterthe background to an enormous array of colors.

W3C (Worldwide Web Consortium), CSS (Cascading Style Sheets) defines CSS as the language describing the presentation of Web pages, colours, fonts and the layouts.CSS allows one to stylize the HTML document according to their need. It contains a set of style rules with which the web pages could be rendered. Not only to that, but it also helps render the page responsively on the devices of different shapes and sizes.

There are two style formatting rules for the CSS. One is called inline styling, and another one is external styling. In the external styling, however, the styling rule sits in a different file. This could then be linked by a unique tag in the HTML document. The cascading part of the CSS refers to how the rules are applied to the HTML elements. The HTML document is styled hierarchically. Therefore, it is the responsibility of CSS to find the precedence of the style rules, accordingly, ultimatelyestablishing a cascading effect.

1.6.3 JavaScript

JavaScript was initially developed to replace the Java Applets which were used in the web browser. In 1996, one of the dominant browsers creating company, Netscape, submitted JavaScript to the Ecma International to influence its presence in all of the browser. Even though each browser has its implementation of the JavaScript,the underlying standard for it is the same. Chrome uses v8 JavaScript Engine; InternetExplorer uses Jscript in contrast to the Mozilla, which uses the standard JavaScript.

JavaScript is a powerful client-side scripting language. JavaScript is employed mainly for enhancing the interaction of a user with the net page. In other words, you can make your web content more lively and interactive, with the assistance of JavaScript. JavaScript is additionally being employed widely in game development and Mobile application development.

Fast forward almost twenty years; JavaScript has become one of the most popular languages for frontend technology. A simple script can be added inline within the HTML document or in a separate file with a link tag in the header file. This way the HTML document knows which script to lead from which location. The main impacting feature of JavaScript is the ability to load or refresh the page content withoutreloading the whole page. This can be achieved by targeting only the concerned tag. Based on the requirement, it also helps on adding the CSS style dynamically. There aremany JavaScript libraries available on the internet, which makes a developer's life easier by leveraging the inner work of JavaScript and help focus on the aesthetic part of it. Some examples are React, Vue.js, jQuery Backbone.js, and Angular.

1.6.4 BootStrap

Bootstrap could be a web framework that focuses on simplifying the event of informative sites. The primary purpose of adding it to an internet project is to use Bootstrap's choices of color, background effect, mobility size, font and layout to it project. As such, the primary factor is whether or not the developers answerable find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML

elements. The result an identical appearance for prose, tables and form elements across web browsers. In addition, developers can cash in of CSS classes defined in Bootstrap to customize the look and component of their tools of their contents. Bootstrap is used for light- and dark-colored tables, more prominent pull quotes, page headings, and text with a highlight.

1.6.5 Python

Python is a general purpose, high level programming language that focuses on the code readability. for web development lines of code will be fewer lines in python compared to other languages. Python is able to do this because of its extensive standard libraries, which make Web development code straightforward and simple. Python maybe used on a variety of systems (Windows,Mac, Linux, Raspberry Pi, etc). Python has a simple syntax that is similar to that of English. Python may be approached in three ways: procedural, object-oriented, and functional. Python is an interpreter language, which means that code may be run as soon as it is written. As a result, prototyping maybe done quickly.

1.6.6 Flask

Flask is a micro-framework designed to create a web application in a short time. It only implements the core functionality giving developers the flexibility to add the feature as required during the implementation. It is a lightweight, WSGI application framework. This framework can either be used for pure backend as well as frontend if need be. The former provides the functionality of the interactive debugger, full request object, routing system for endpoints, HTTP utilities for handling entity tags, cache controls, dates, cookies etc. It also provides a threaded WSGI server for local development including the test client for simulating the HTTP requests. Werkzeug and Jinja are the two core libraries The Jinja, however, is another dependency of the Flask. It is a full-featured template engine. Sandboxed execution, powerful XSS prevention, template inheritance, easy to do debug, configurable syntax is it's few of many features. In addition, the code written in the HTML template is compiled as python code.

Since Flask is often termed as a prototyping framework, it does not include the abstraction layer for the database or any sorts of validation and security whatsoever.

Therefore, Flask has given full flexibility to the implementor to add the requirements. There are extensions available for the Flask frameworks. Libraries but not limited to are, gunicorn for server, SQLAlchemy for database, Alembic for database migration management, celery & Redis for an asynchronous task runner, Flask-WTF form for form validation and Flask-limiter for rate-limiting the web requests. Flask is available for Python 3 and the newer version; it is also available in PyPy and easily installable with Python's official package manager pip.

Flask is a Python API that allows us to create web-based applications. It was developed by Armin Ronacher. Flask's framework is easier to understand than Django's, and it takes less fundamental code to develop a simple web application. A Web-Application Framework, often known as a Web Framework, is a set of modules and libraries that allow programmers to create apps without having to write low-level code such as protocols and thread management. Flask is an open-source web framework.

This paper is structured as follows: Chapter 2 introduces the Literature Survey, Chapter 3 describes about the requirements for developing the project, Chapter 4 gives an overview of Methodologies that are implemented, Chapter 5 shows the system design, Chapter 6 discusses about the Results and Discussions. Finally Chapter 7 shows the Conclusion and Future Scope.

Chapter 2

LITERATURE SURVEY

2.1 Study of Related Papers

Paper [1] Jiang, J., Zhang, H., Dai, C., Zhao, Q., Feng, H., Ji, Z., & Ganchev, I. (2021). Enhancements of attention-based bidirectional lstm for hybrid automatic text summarization. IEEE Access, 9, 123660-123671.

In this paper research has put forward enhancements to the structure of the attention-based bi-directional LSTM model ('Bi-LSTM + Attention') and the attention-based sequence model ('Seq2Seq + Attention') in order to improve the Automatic Text Summarization (ATS). Firstly, a novel enhanced semantic network (ESN) model has been proposed, which works out the semantic similarity between the encoder and decoder, and maximizes the semantic relevance during training, which increases the probability of generating more accurate text summaries, thus also improving the correlation with the source text. Secondly, aiming at solving the problem of unregistered words, a novel 'DA-PN' model has been proposed, which utilizes decoder attention (DA) based on a pointer network (PN). Thirdly, it has been proposed to combine the elaborated 'DA-PN' model with a coverage mechanism integrating multi-attention.

Table 2.1: Performance comparison of models

Model	ROUGE-1	ROUGE-2	ROUGE-L
'Bi-LSTM + Attention' (baseline)	25.27	8.03	29.04
'Seq2Seq + Attention' (baseline)	27.07	9.03	30.56
ESN	**29.17**	**10.97**	**30.95**
'DA_PN'	**32.20**	**13.46**	**31.26**
'DA-PN + Cover'	**33.43**	**13.77**	**32.93**
'DA-PN + Cover + MLO'	**33.75**	**14.09**	**32.69**

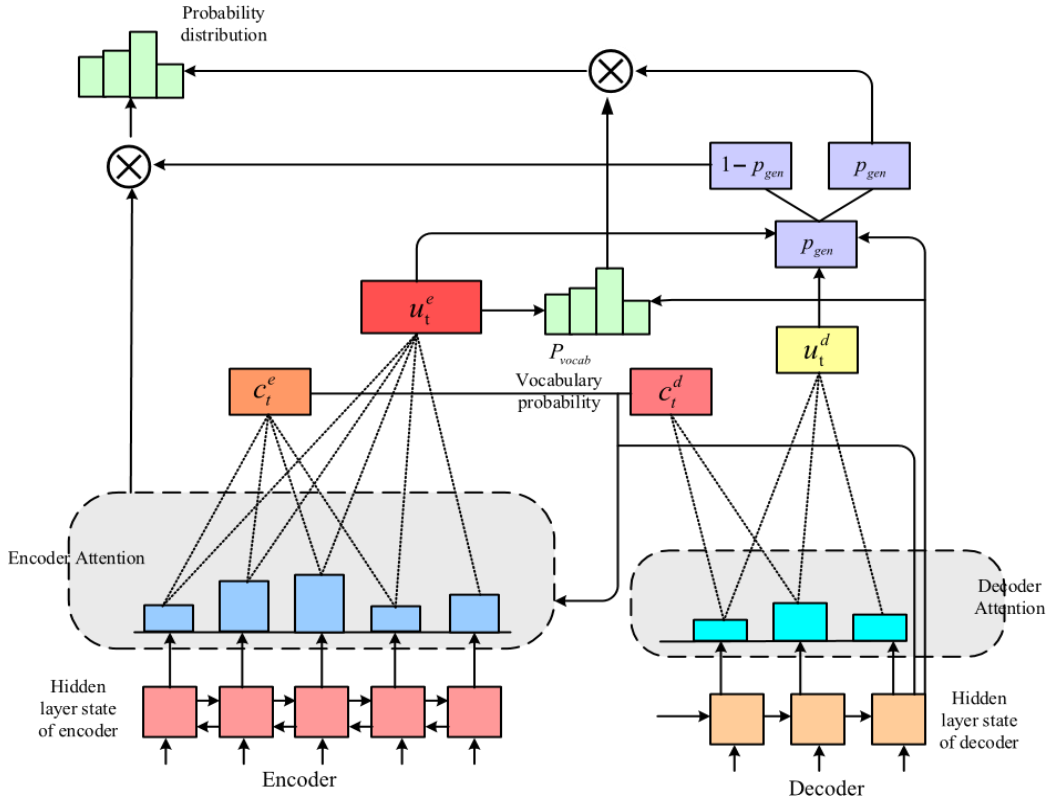

Figure 2.1: The proposed 'DA-PN + Cover' model.

In the resultant 'DA-PN + Cover' model, by using the attention distribution of the encoder and decoder in all the previous time steps, the attention of the current time step is affected in a positive way, leading to finding more accurate words for inclusion in the text summaries by avoiding repeated words.

Lastly, in order to prevent the spread of cumulative errors in generated text summaries, it has been proposed to add a mixed learning objective (MLO) function to the 'DA-PN + Cover' model. The resultant 'DA-PN + Cover + MLO' model is the best performing one among the ATS models proposed in this paper.

Paper [2] Alami, N., Meknassi, M., & En-nahnahi, N. (2019). Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. Expert systems with applications, 123, 195-211.

In this paper, due to the large number of unlabeled data sets available and not sufficient label data to train supervised models, unsupervised approaches to DL are preferable. Since a variety of unsupervised approaches to DL have been introduced to learn features from unlabeled data sets, the problem of a lack of data sets labelled is no anymore a problem. Auto-Encoder (AE), Variational Auto Encoder (VAE), and ELM-AE are a couple of illustrations of such models used in this study. Instead of using the Bag of Words representation in this paper, word2vec model utilized as the input for models.

In addition, they unveiled a brand-new model based on ELM-AE for text summarization. Both the BOW and word2vec approaches used to training the model ELM-AE were studied for their effects. First of all, the output is always equal to the output in any auto encoder model. Finding the latent space is the problem at hand, though (compressed data). The decoder model is altered based on the error parameters. Mean square error is the error measure in use here. The Gaussian distribution with mean and SD as parameters must be utilized for the latent space in the variational auto encoder model. These parameters are used to produce the outcome.

In extreme learning machine auto encoder the training parameters are adjusted based on three conditions they are:

$$\beta = \left(\frac{I_M}{C} + H^T H\right)^{-1} H^T X$$

$$\beta = H^T \left(\frac{I_N}{C} + H H^T\right)^{-1} X$$

$$\beta = H^{-1} X$$

Finally, we propose three new ensemble techniques that combine the results provided by different investigated models through a voting technique.

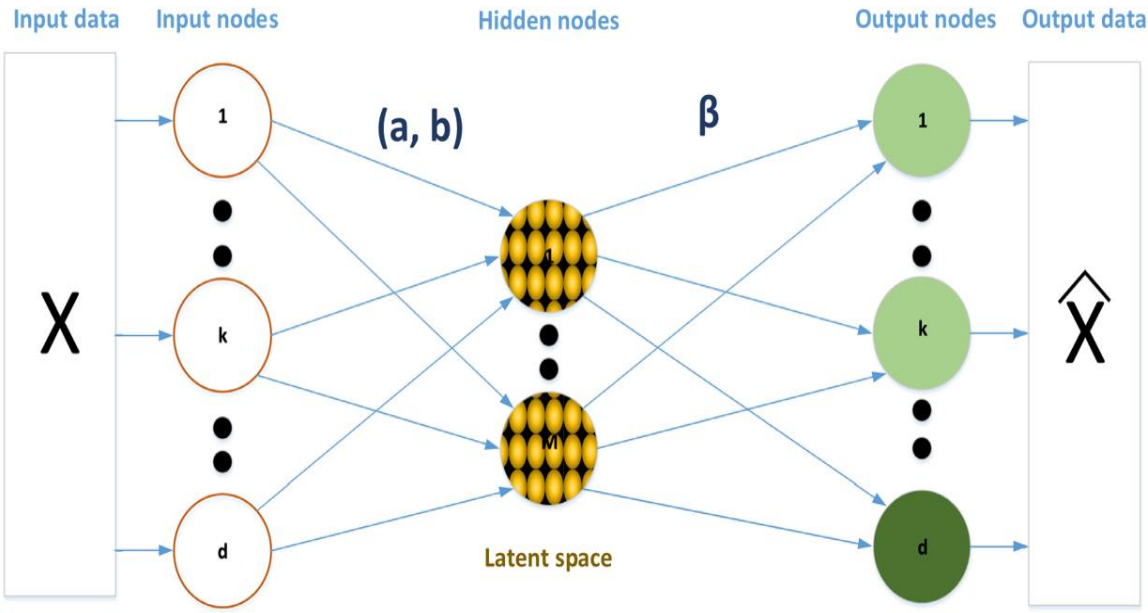

Figure 2.2: ELM-AE model

d > M: To compress latent space

d = M: Equal input and latent space dimensions

d < M: Enlarged representation

Paper [3] Zhao, S., You, F., & Liu, Z. Y. (2020). Leveraging Pre-Trained language model for summary generation on short text. IEEE Access, 8, 228798-228803.

In this paper a short text summary generation model is proposed based on keyword templates and improve the data preprocessing method of Chinese short text in summary generation tasks. a model for creating short text summaries based on keyword templates is put forth in an effort to enhance how Chinese short texts are preprocessed for use in these activities.They used the classic encoder-decoder structure and tried to increase effectiveness of their suggested method in producing brief text summaries. The decoder uses eight Transformers stacked with random initialization, whereas the encoder uses BERT for initialization. Their model was given the name BSA. If the model divides sentences into categories using keywords, it is called BSA*. They demonstrated how to effectively construct brief text summaries using the pre-trained language model, and they demonstrated how to validate it using the abstractive method..

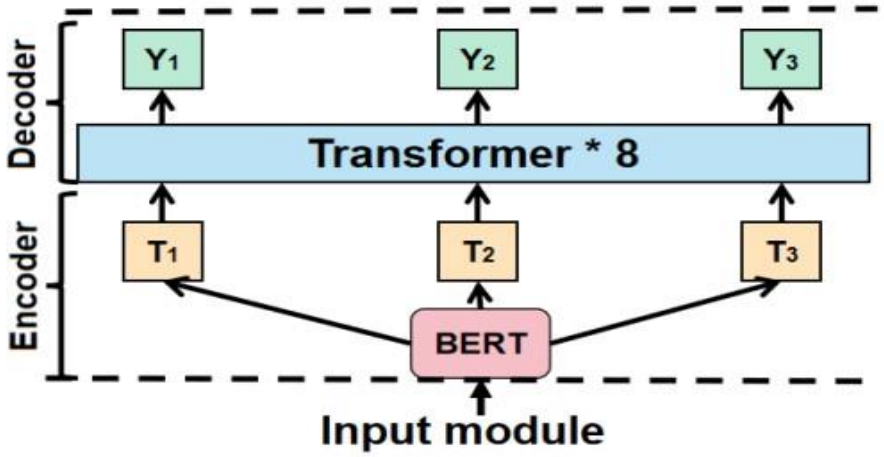

Figure 2.3: The BSA model's architectural structure. It includes a Decoder (BERT) and an Encoder (Transformer)

This model can function as a springboard to raise the summary's quality and make the pre-trained language model more effective at producing concise text summaries. In order to train on a big corpus, the BERT model employs "masked language modelling" and "next sentence prediction" techniques. BERT is useful for an amount of downstream tasks in natural language processing since when used on them, it can be tweaked to give exceptional results.

The proposed model achieves a better performance with 44.2 F-score of ROUGE-1, 28.9 ROUGE-2 and 39.2. Compared with the RTCS model, proposed model is improved by 4.3, 7.4, and 1.3.

Paper [4] You, F., Zhao, S., & Chen, J. (2020). A topic information fusion and semantic relevance for text summarization. IEEE Access, 8, 178946-178953.

In Text summarization is to make the text easier to read and understand, especially for poor readers. Features of Website. There are two kinds of summary generation: extractive and abstractive. Extractive summarization models extract some important words or sentences from the original documents. Compared with abstractive methods, it is comparatively mature and straight forward. The model compresses the original document into dense vectors with the encoder, and the decoder generates summary using the compressed vectors. A high-quality summarization system needs to focus on the topic content of the document and the

similarity between the summary and the source document. They propose a topic information fusion and semantic relevance for text summarization based on Fine-tuning BERT. Firstly, Topic keywords are extracted and fusion them with source documents as part of the input. Secondly, aiming the summary closer to the source document by calculating the semantic similarity between the generated summary and the source document, the quality of the abstract is improved.

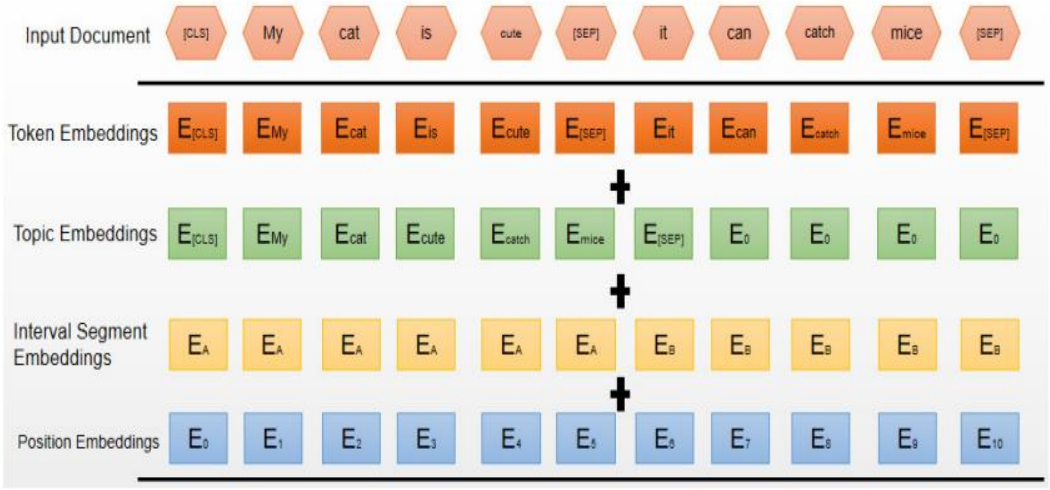

Figure 2.4: The TIF-SR model's input structure.

Calculating the semantic similarity between the generated summary and the original document, and maximize the similarity score through training. The experimental data show that the proposed model has acquired good results.

Paper [5] Joshi, A., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019). SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders. Expert Systems with Applications, 129, 200-215.

In this paper, Text summarizing techniques come in two different varieties. Includes extractive and abstractive summary. In this study, Summary of general extracted text is done using auto-encoders. The query-focused summarization, in contrast to the general summarization, demonstrates the document's key contents in accordance there with user-provided queries. Before sending each document to the sentence encoder's next stage, we first pre-processed it. Second, the objective of an encoder network is to transform an English sentence into a vector, which is then attempted to be translated into nearby

sentences by the decoder. The connection between a source and a destination is made possible by the encoder-decoder design, which forces the model to pick out and abstract some crucial aspects. In order to map phrases with comparable semantic and syntactic properties into identical vector representations, the entire network is trained to reconstruct the surrounding sentences. An auto-encoder is a specific kind of feed-forwarded neurons system developed by Hinton and Salakhutdinov (2006) to minimize data dimensionality. The input and output layers are identical, along with at least single secret layer with smaller dimensions than the input data is included. We only maintained the encoder portion of the auto-encoder network after training it, discarding the decoder portion, to produce a lower dimensional representation of each textual-unit embedding. The lower-dimension encoder output was referred to as "Latent Representation." This hidden representation of the entire text is merely a synopsis of it

Paper [6] JUGRAN, S., KUMAR, A., TYAGI, B. S., & ANAND, V. (2021, March). Extractive automatic text summarization using SpaCy in Python & NLP. In 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 582-585). IEEE.

In this paper author described as in the digital world, as the amount of data produced at every instance is very huge .There is an ultimate need to develop a machine that can reduce the length of the texts automatically. Applying text summarization gears up the procedure of researching, reduce reading time and increase the amount of information generated in the specific field. In this project, they tried to create a model as the solution which is based on extractive approach for summarization text, starting with NLP as the fundamental model. The extractive approach is actually successful in delivering the summery using the set of words which is actually most improve words in the actual text, hence the relevant information.

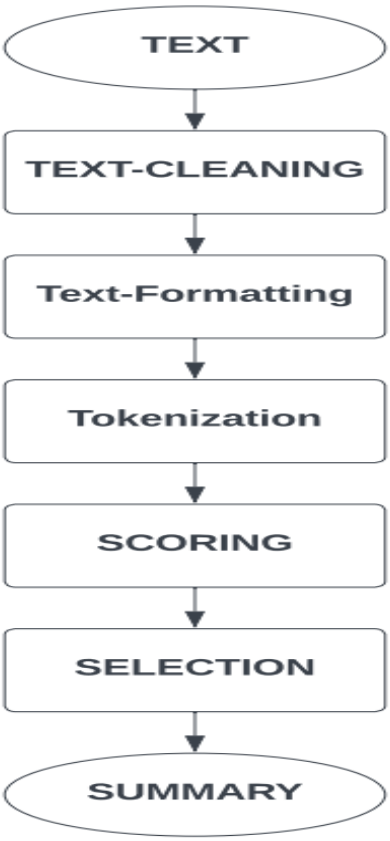

Figure 2.5: Step-by-step Implementation

Existing model were created based on NLTK that is a library for processing text string by string. The input and output using NLTK is the sequence of characters that is string. Provide various algorithm for a particular problem is the time consuming. The proposed model utilizes the library Spacy which selects the best option itself becoming more efficient.

Paper [7] L. Zhang, Y. Yang, J. Zhou, C. Chen and L. He, "Retrieval-Polished Response Generation for Chatbot," in IEEE Access, vol. 8, pp., 2020.

In this paper author(s) described about the Chatbot communication, an essential task in both natural language processing and artificial intelligence fields, involves a robot communicating with human beings using natural language in open domains. Chatbots are able to understand the intent of the conversation rather than just use the information to communicate and respond to queries. Business owners are starting to feed their chatbots with actions to “help”

them become more humanized and personal in their chats. Chatbots have, and will always, help companies automate tasks, communicate better with their customers and grow their bottom lines. But, the more familiar consumers become with chatbots, the more they expect from them.

With personalization being the primary focus, you need to try and "train" your chatbot about the different default responses and how exactly they can make customers' lives easier by doing so. With NLP, your chatbot will be able to streamline more tailored, unique responses, interpret and answer new questions or commands, and improve the customer's experience according to their needs.

Chatbots play a critical role in many real-world applications, such as smart speakers, customer service systems, and social robots. Research on chatbots began in the 1960s initially, researchers used sets of handwritten rules and templates. However, such rule-based models require significant human effort and lack flexibility. In recent years, as large-scale dialog corpora and high-speed computational resources have become available, the early rule-based models have been rapidly replaced by data-driven models. The existing data-driven models can be categorized as retrieval-based or generation based.

Chapter 3

REQUIREMENT SPECIFICATIONS

3.1 Functional Requirements

- The website should work on all the devices without any interruption in the services.
- The website should allow all users to get authenticated and verified with theirEmails
- Website uses the Internet to connect with server and displays the products to the user.
- Website will store the basic data of the User information like Name, Email ID,Phone, *isUserSignedIn* Status.
- Details which are entered by the user should be correct else the informationwhich is displayed may not be accurate.
- Users can communicate to customer service through Chatbot directly from the Website itself.
- Customers can add the selected products to the cart and order the items with payment options.
- Several Payment options are provided to the user like Credit card, Debit cards,UPI payments etc.
- Users can log out from the account anytime from the website with the help ofSign out Button.
- Users can sign in to their account from any device and their basic data getsloaded.
- Only one time sign in, users are not required to sign in every they use the website.

3.2 Non-Functional Requirements

- Availability
- Performance
- Reliability
- Uptime of servers
- The website start time should be within less than 10 seconds

- Website should work efficiently
- The list of products displayed should be correct and no incorrect details shouldbe shown.
- Website should be given basic permission like INTERNET, and STORAGE for uninterrupted use of the app.
- The data should fetch from the server within less than 3 seconds and there shouldbe no any delay while retrieving the data.
- Usability: Regardless of the size of your business, the website of your businessshould be easy to use for even a non-technical user.
- App size is minimized to make the app run efficiently and accurately on theuser's device.

3.3 Software Requirements

3.3.1 Frontend Frameworks:

- HTML
- CSS
- BootStrap
- JavaScript

3.3.2 Backend Frameworks:

- Python
- Flask
- Twitter API
- Web API

3.4 Hardware Requirements

3.4.1 Windows Version:

- 64-bit Microsoft® Windows® 8/10
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPUwith support for a Windows Hypervisor
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Anaconda or Jupyter Notebook)
- 1280 x 800 minimum screen resolution

Chapter 4

METHODOLOGY

4.1 Accessing the data

4.1.1 Web data

Web data is the data that comes from large or diverse number of sources. Web data are developed with the help of Semantic Web tools such as RDF, OWL, and SPARQL. Also, the web data allows sharing of information through HTTP protocol or SPARQL endpoint.

There are two ways to extract data from a website:

- Using API of the website
- Using web Scrapping

In our project we will focus on web Scrapping. Web scraping is the process of collecting and parsing raw data from the Web, and the Python community has come up with some pretty powerful web scraping tools. The Internet hosts perhaps the greatest source of information—and misinformation—on the planet. Many disciplines, such as data science, business intelligence, and investigative reporting, can benefit enormously from collecting and analyzing data from websites. This is implemented using a python framework called Beautiful Soup.

4.1.2 Twitter data

Twitter data provides you with a snapshot of your Twitter information, So presented broadly, “Twitter data” is more accurately described as semi-structured data because we have a mixture of structured data and unstructured data to observe; but the tweet text in and of itself is unstructured data. To access the Twitter data Signing up for a developer account is quick and easy, just answer a few questions and you can start exploring and building on the Twitter API v2 using Essential access. Next you will create a Project and an associated developer App during the onboarding process, which will provide you a set of credentials that you will use to authenticate all requests to the API. Twitter access keys and tokens will only display once in the developer portal, so it is important that you store these credentials in your password management system as soon as you generate them. The tweets are extracted

from twitter using twitter API using Tweepy library in python. Every tweet in twitter consists of an identifier and these tweets id's are utilized to extract them.

4.2 Data Pre-Processing

Preprocessing is a crucial step in the processing of text. A text can comprise of words, sentences and1 paragraphs. A meaningful sequence of characters is considered as text. The most widely used text preprocessing techniques are available in the NLTK (Natural Language Tool Kit) library in python. You must preprocess the text. It includes splitting the text into sentences, lowering the case of all words, removing stop words (is, an, the, etc.) and punctuation and other tasks. The goal is not to waste resources (compute power, time) processing things that don't add much value to extracting the semantics and understanding the text. Specifically to identifying and splitting the text into sentences, it's crucial as later the algorithm will score and select them to be part of the summary.

In text summarization or tweets summarization the main techniques for data preprocessing are stop ward removal, text cleaning, stemming, lemmatization, POS tagging, Tokenization. Unlike other methods these methods are very use full handling text data. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP).

Lemmatization is a text normalization technique used in Natural Language Processing (NLP), that switches any kind of a word to its base root mode. Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning.

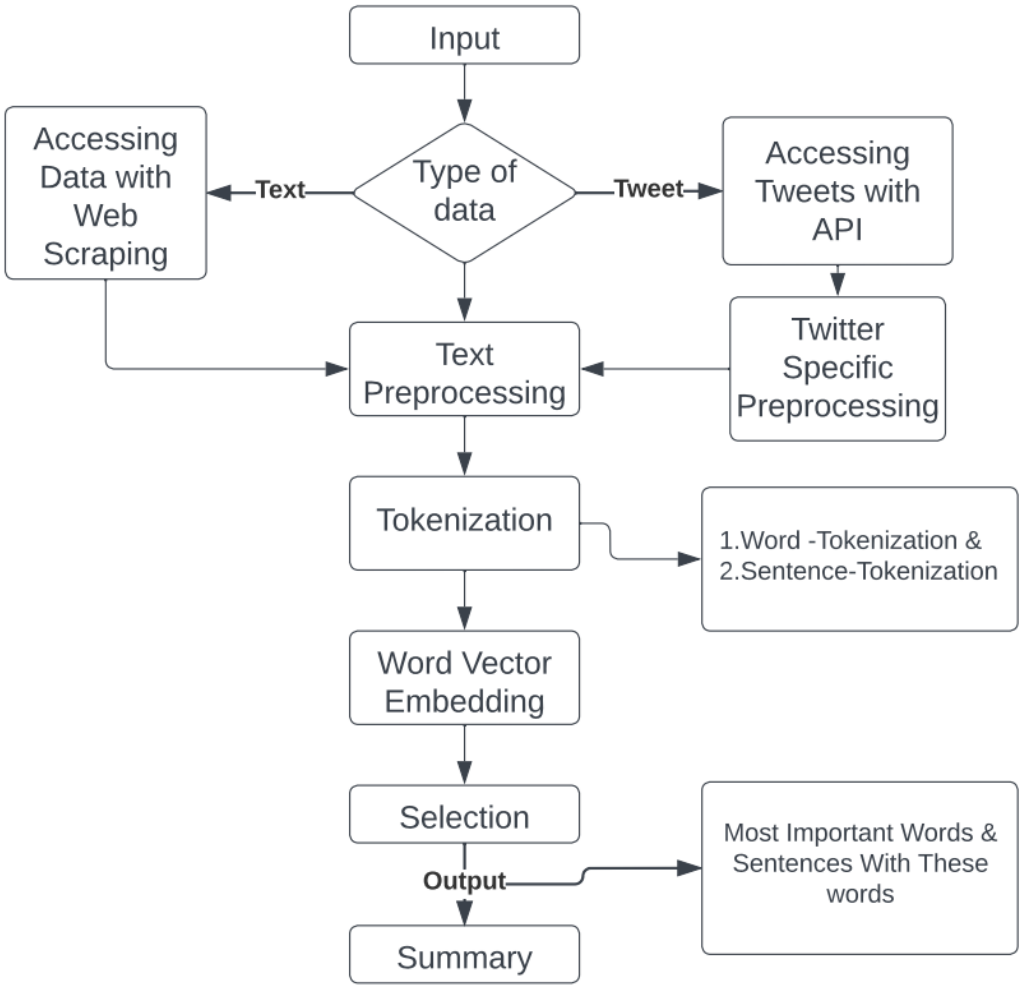

Figure 4.1: Auto Text Summarizer Architecture

4.2.1 Text Cleaning

Text cleaning is the process of preparing raw text for NLP (Natural Language Processing) so that machines can understand human language. The goal of data prep is to produce 'clean text' that machines can analyze error free. Clean text is human language rearranged into a format that machine models can understand. Text cleaning can be performed using simple Python code that eliminates stopwords, removes unicode words, and simplifies complex words to their root form. Lowercasing the data, Removing Punctuations, Removing Numbers, Removing extra space, Replacing the repetitions of punctuations, Removing Emoji's.

4.2.2 POS Tagging

It is a process of converting a sentence to forms – list of words, list of tuples. The tag in case of is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, and so on. Default tagging is a basic step for the part-of-speech tagging. It is performed using the DefaultTagger class. The DefaultTagger class takes 'tag' as a single argument. NN is the tag for a singular noun. Default Tagger is most useful when it gets to work with most common part-of-speech tag. That's why a noun tag is recommended. Except stop words all the words in the sentence are tagged with respective to its parts of speech like noun, verb and pronoun etc.

4.2.3 Stop Word Removal

Our Stop word removal is one of the most commonly used preprocessing steps across different NLP applications. The idea is simply removing the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally classified as stop words. These words have no significance in some of the NLP tasks like information retrieval and classification, which means these words are not very discriminative. On the contrary, in some NLP applications stop word removal will have very little impact. Most of the time, the stop word list for the given language is a well hand-curated list of words that occur most commonly across corpuses. While the stop word lists for most languages are available online, the stop words are those words that are less helpful in the further analysis of the tweet. These words removed from the tweets before fed to the algorithm.

4.3 Tokenization

Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph. The tokens could be words, numbers or punctuation marks. In tokenization, smaller units are created by locating word boundaries. Before processing a natural language, we need to identify the words that constitute a string of characters. That's why tokenization is the most basic step to proceed with NLP (text data). This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

Methods to perform Tokenization:

4.3.1 Word Tokenization

A word-based tokenization algorithm will break the sentence into words. The most common one is splitting based on space. This is the most commonly used tokenization technique. It splits a piece of text into words based on a delimiter. The most commonly used delimiter is space. You can also split your text using more than one delimiter, like space and punctuation marks. Depending on the delimiter you used, you will get different word-level tokens.

When the entire text is divided into individual words and word-score is generated for every word according to its count. Word-based tokenization can be easily done using custom RegEx or Python's split() method. Apart from that, there are plenty of libraries in Python — NLTK, spaCy, Keras, Gensim, which can help you perform tokenization easily.

Example: "Is it weird I don't like coffee?"

By performing word-based tokenization with space as a delimiter, we get:

["Is", "it", "weird", "I", "don't", "like", "coffee?"]

4.3.2 Sentence Tokenization

Sentence tokenization is the process of splitting text into individual sentences. For literature, journalism, and formal documents the tokenization algorithms built in to spaCy perform well, since the tokenizer is trained on a corpus of formal English text. The sentence tokenizer performs less well for electronic health records featuring abbreviations, medical terms, spatial measurements, and other forms not found in standard written English.

When the entire text is divided into individual sentences and each sentence is provided to its sentence-score according to the occurrence of the high-scored words. ClarityNLP attempts to improve the results of the sentence tokenizer for electronic health records. It does this by looking for the types of textual constructs that confuse the tokenizer and replacing them with single words. The sentence tokenizer will not split an individual word, so the offending text, in replacement form, is preserved intact during the tokenization process. After generating the individual sentences, the reverse substitutions are made, which restores

original text in a set of improved sentences. ClarityNLP also performs additional fixups of the sentences to further improve the results.

4.4 Word Embedding

Word Embedding is an approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. Word to vector is an important feature of a language model. It gives you insights into a language model and how effective that model is. There is no general w2v (word to vector) model for a language, you want to use. You need to build it for your language depending on the domain and usage for specific applications. You can build a large w2v model but it creates a heavy burden of performance and it will use a lot of resources in the applications. So, we need to choose the dataset accordingly and build a w2v model for that specific application. For example, if you are building a model for news, choose the news dataset and then build a w2v model using that dataset.

GloVe:

1. GloVe (Global Vectors for Word Representation) word embedding are vector representation of words.
2. These word embedding will be used to create vectors for our sentences. We could have also used the Bag-of-Words or TF-IDF approaches to create features for our sentences, but these methods ignore the order of the words (and the number of features is usually pretty large).
3. We will be using the pre-trained Wikipedia 2014 + Gigaword 5 GloVe vectors.

4.5 Sentence Selection

Automatic Text Summarization is one of the most challenging and interesting problems in the field of Natural Language Processing (NLP). It is a process of generating a concise and meaningful summary of text from multiple text resources such as books, news articles, blog posts, research papers, emails, and tweets. The demand for automatic text summarization systems is spiking these days thanks to the availability of large amounts of textual data. Sentence selection is the crucial step for summarization. For sentence selection we will be using some python libraries and NLP techniques, which will help in summarization.

4.5.1 NLTK based method

NLTK, or Natural Language Toolkit, is a Python package that you can use for NLP. It is an essential library supports tasks such as classification, stemming, tagging, parsing, semantic reasoning, and tokenization in Python. It's basically your main tool for natural language processing and machine learning. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project. Natural Language Processing with Python provides a practical introduction to programming for language processing.

4.5.2 spaCy based method

spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python. It is designed specifically for production use. It will be used to build information extraction, natural language understanding systems, and to pre-process text for deep learning. It supports deep learning workflow in convolutional neural networks in parts-of-speech tagging, dependency parsing, and named entity recognition.

Frequency of words is calculated and stored in a dictionary. The sentences having high frequency words will be having high priority, combination of these sentences is summary.

4.5.3 Text Rank Algorithm

Before getting started with the Text Rank algorithm, there's another algorithm which we should become familiar with – the PageRank algorithm. In fact, this actually inspired Text Rank! PageRank is used primarily for ranking web pages in online search results. The pages contain links pointing to one another. Some pages might have no link – these are called dangling pages. In order to rank these pages, we would have to compute a score called the PageRank score. This score is the probability of a user visiting that page. To capture the probabilities of users navigating from one page to another,

we will create a square matrix M, having n rows and n columns, where n is the number of web pages. Each element of this matrix denotes the probability of a user transitioning from one web page to another.

The initialization of the probabilities is explained in the steps below:

- Probability of going from page i to j, i.e., M[i][j], is initialized with 1/(number of unique links in web page wi)
- If there is no link between the page i and j, then the probability will be initialized with 0
- If a user has landed on a dangling page, then it is assumed that he is equally likely to transition to any page. Hence, M[i][j] will be initialized with 1/(number of web pages)

Finally, the values in this matrix will be updated in an iterative fashion to arrive at the web page rankings.

Let's understand the Text Rank algorithm, now that we have a grasp on PageRank. I have listed the similarities between these two algorithms below:

- In place of web pages, we use sentences
- Similarity between any two sentences is used as an equivalent to the web page transition probability
- The similarity scores are stored in a square matrix, similar to the matrix M used for PageRank

Text Rank is an extractive and unsupervised text summarization technique. Let's take a look at the flow of the Text Rank algorithm that we will be following:

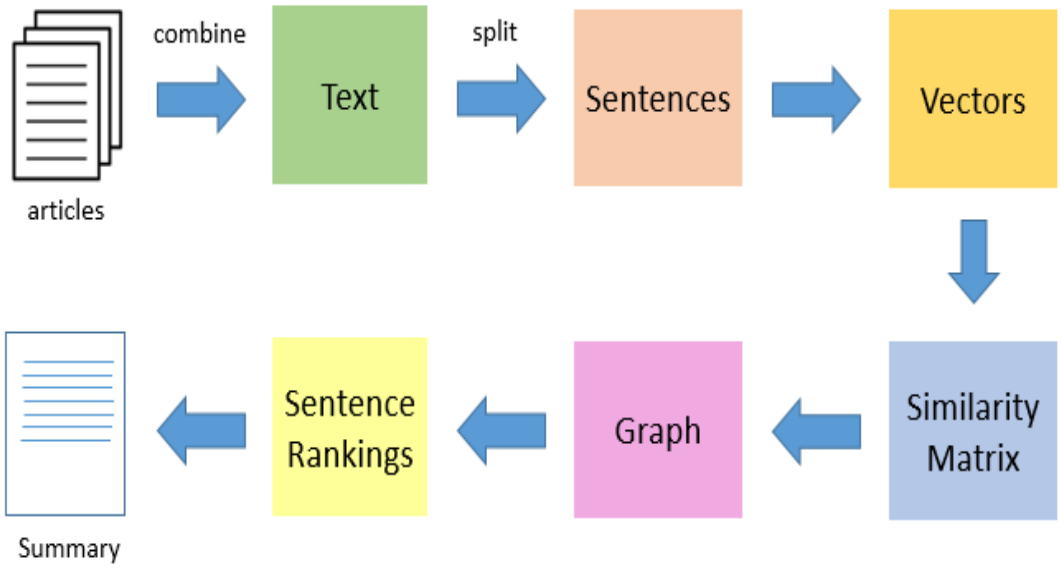

Figure 4.2: Text Rank Algorithm

Generate sentence vector embedding: Using the word vector embedding, generate sentence vector embedding by averaging the sum of word vectors.

It is calculated as W / I. Where,

W = Vector sum of words in the sentence

I = Length of the sentence

Create similarity matrix: Similarity matrix represents the similarity between each sentence with every other sentences.

Determine sentence rank: It is determined graphically by using page rank algorithm.

Generate Summary: Generate a summary by extracting the sentences having higher ranks.

4.5.4 T5 Transformer based method

T5 (Text-To-Text Transfer Transformer) is a transformer model that is trained in an end-to-end manner with text as input and modified text as output, in contrast to BERT-style models that can only output either a class label or a span of the input. This text-to-text formatting makes the T5 model fit for multiple NLP tasks like Summarization, Question-Answering, Machine Translation, and Classification problems.

Google's T5 is one of the most advanced natural language models to date. It builds on top of previous work on Transformer models in general. Unlike BERT, which had only encoder blocks, and GPT-2, which had only decoder blocks, T5 uses both. T5 expects a prefix before the input text to understand the task given by the user. For example, "summarize:" for the summarization, "cola sentence:" for the classification, "translate English to Spanish:" for the machine translation, etc.

4.6 Integrating Website with the Backend

User should need a home page where they can access the summarization tool. So a well-organized database is very essential for maintenance of an summarization tool. Various tools need to be accessed from home page are twitter hashtag summarization, short text summarization, article summarization through URL as input from the user. When the user gives twitter hash tag as input in the form, this will be given as input to python file "access_data.py". In this python file connection with twitter developer account is established and tweets from this account can be accessed using twitter API.

All the accessed tweets are written into a separate file "actual_data.py". This tweets are further preprocessed using regular expression and these cleaned tweets are written into another file called "cleaned_data.py". This cleaned data is given to text Rank algorithm, NLTK and T5 Transformer model to get the required output (Summary).

While coming to the web data the URL is given as input by the user. Here the requirement is the data present in the website need to be summarized. This access of data is done using web scrapping in python. This url summarization can be accessed from the navigation present in home page. The URL routing in this web data summarization will be as '/article'.

Front end made by Scripting languages like HTML,CSS3,JavaScript and Bootstrap .This markup language make the website more attractive and useful and user-friendly to use and summarize .Markup languages help in making the things more attractive and imaginary.

Chapter 5

SYSTEM DESIGN

5.1 UML Diagrams

5.1.1 Activity Diagram

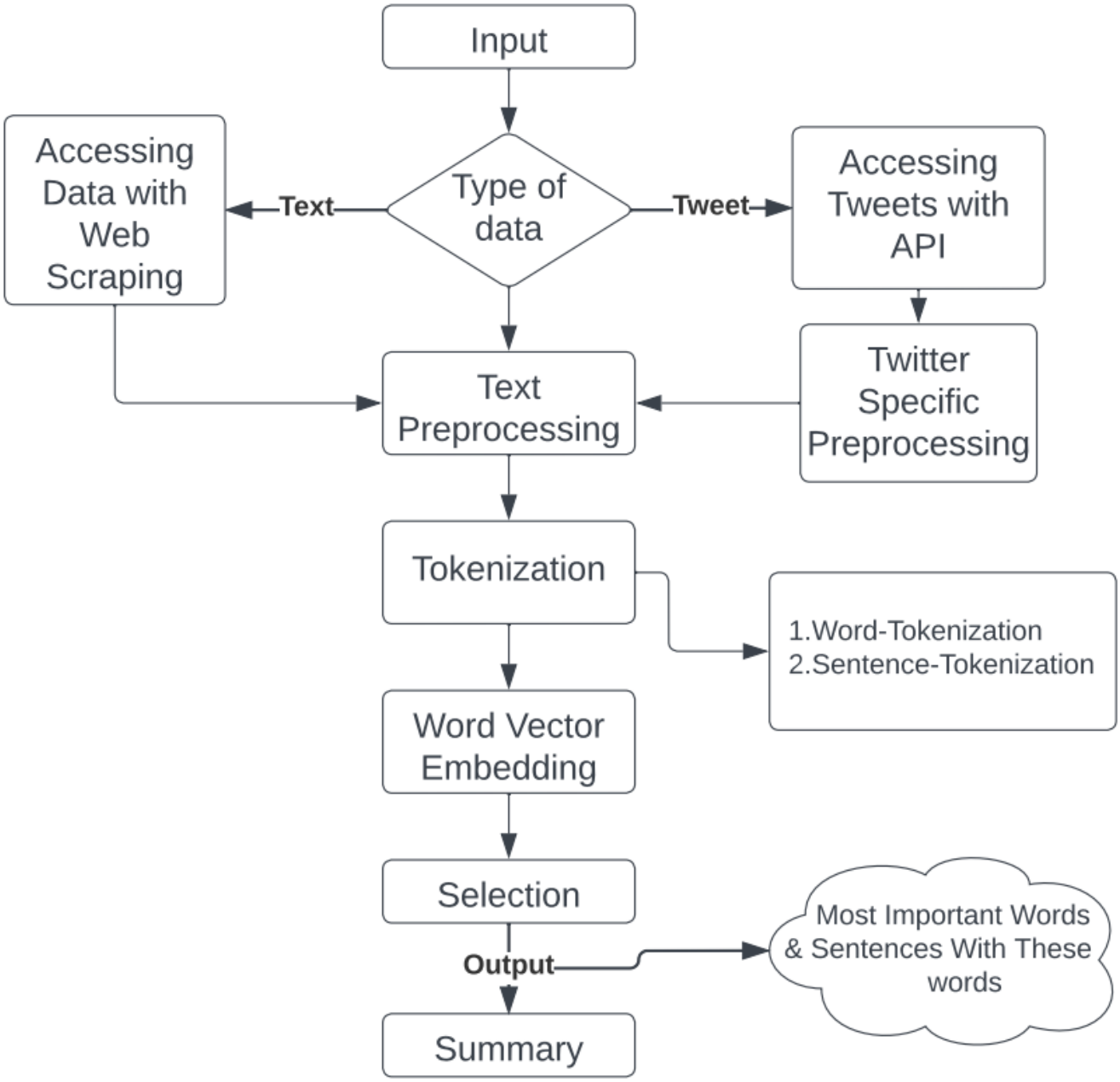

Figure 5.1 Activity Diagram

5.1.2 Use case diagram

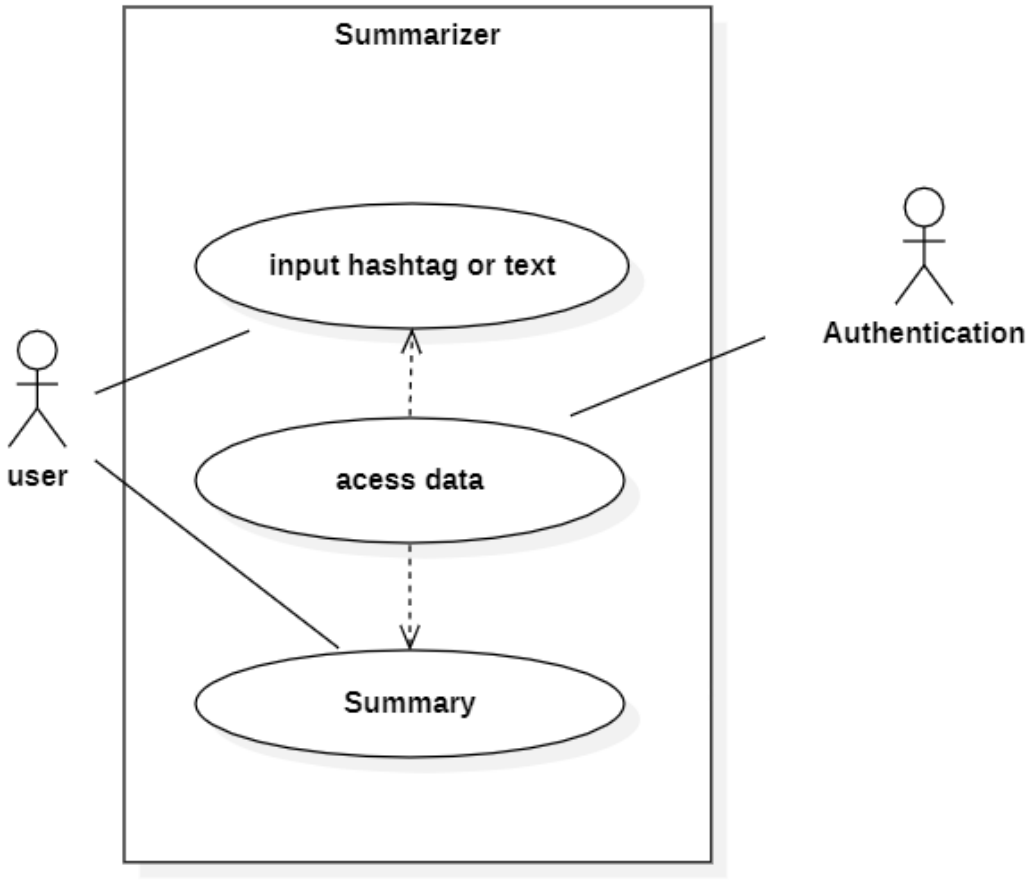

Figure 5.5 Use case diagram

5.1.3 State Chart Diagram

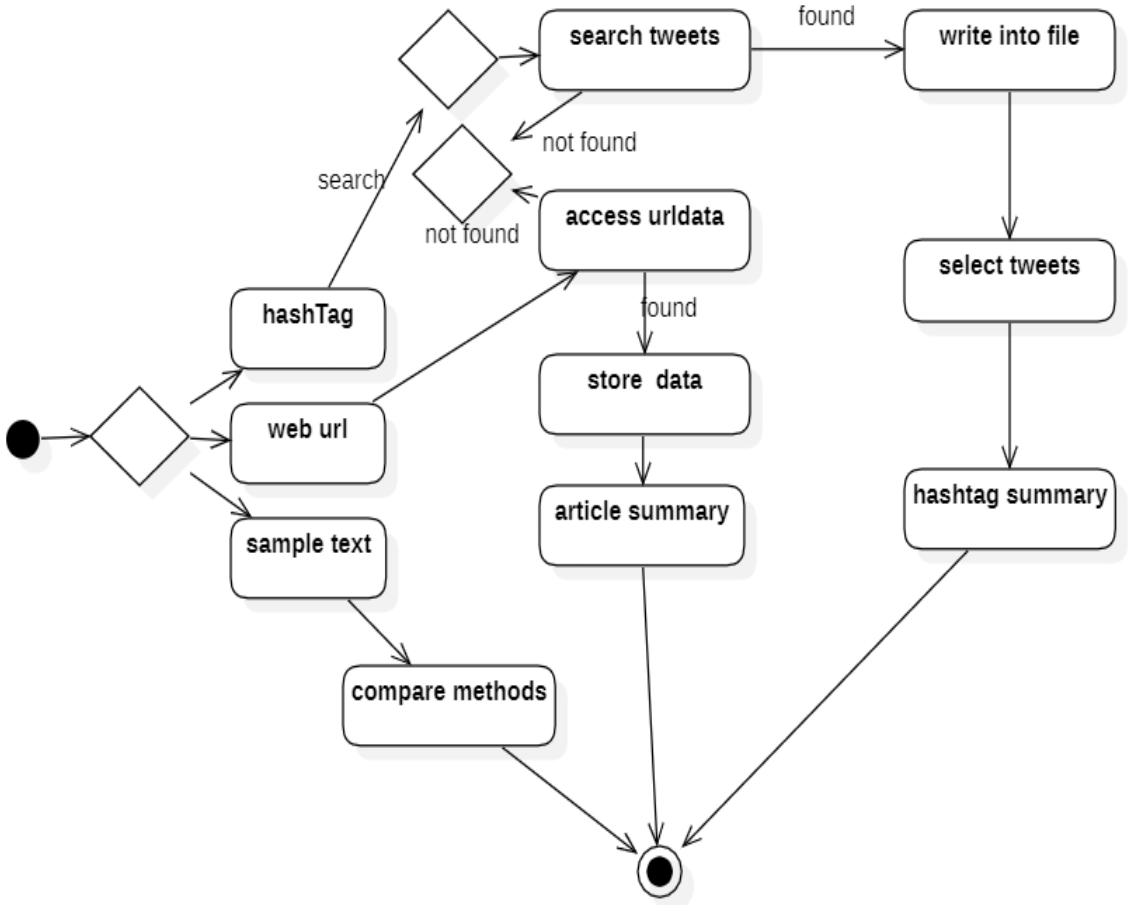

Figure 5.6 State Chart Diagram

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Dataset

In this project we used the CNN/DailyMail non-anonymized summarization dataset. There are two features:

Article: text of news article, used as the document to be summarized.

Highlights: joined text of highlights with and around each highlight, which is the target summary.

- **Homepage:** https://github.com/abisee/cnn-dailymail
- **Source Code:** https://github.com/tensorflow/datasets/tree/master/tensorflow_datasets/summarization/cnn_dailymail.py
- **Versions:**
  - 1.0.0: New split API (https://tensorflow.org/datasets/splits)
  - 2.0.0: Separate target sentences with newline. (Having the model predict newline separators makes it easier to evaluate using summary-level ROUGE.)
  - 3.0.0: Using cased version.
  - 3.1.0: Removed BuilderConfig
  - 3.2.0: Remove extra space before added sentence period. This shouldn't affect ROUGE scores because punctuation is removed.
  - 3.3.0: Add publisher feature.
  - 3.4.0 (Default): Add ID feature.
- **Splits**:

Table 6.1: Split Dataset

Split	Examples
'test'	11,490
'train'	287,113
'validation'	13,368

6.2 Testing the model with the Dataset

The proposed model is tested using cnn_dailymail dataset by calculating the ROUGE-1, ROUGE-2 & ROUGE-L metrics.

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially a set of metrics for evaluating automatic summarization of texts as well as machine translations. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced).

In addition, ROUGE scores are branched into ROUGE-1, ROUGE-2 and ROUGE-L scores.

Formula for measuring NLP accuracy with ROUGE,

$$\frac{\text{number of n-grams found in model and reference}}{\text{number of n-grams in } \textbf{model}}$$

ROUGE-1: Precision and Recall compare the similarity of uni-grams between reference and candidate summaries.

ROUGE-2: Precision and Recall compare the similarity of bi-grams between reference and candidate summaries.

ROUGE-L: Precision and Recall measures longest matching sequence of words using LCS. An advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order. Since it automatically includes longest in-sequence common n-grams, you don’t need a predefined n-gram length.

Table 6.1: Performance of Proposed model

Model	ROUGE-1	ROUGE-2	ROUGE-L
'Bi-LSTM + Attention' (baseline)	25.27	8.03	29.04
'Seq2Seq + Attention' (baseline)	27.07	9.03	30.56
ESN	**29.17**	**10.97**	**30.95**
'DA_PN'	**32.20**	**13.46**	**31.26**
'DA-PN + Cover'	**33.43**	**13.77**	**32.93**
'DA-PN + Cover + MLO'	**33.75**	**14.09**	**32.69**
TextRank (Proposed Algorithm)	**32.90**	**18.02**	**18.03**

6.3 Comparative Study

The table below offers a brief comparison of the various approaches in NLP used in the summary process in order to summarize the text using adaptive models. This is a summary analysis that was produced from research that was mostly focused on NLP approaches and was published in a variety of magazines.

Table 6.2: Comparison of different Techniques used in Summarization of the Text.

	Author	Approach	ROUGE SCORE		
			Rouge-1	Rouge-2	Rouge-L
Paper-1	Jiawen Jiang, Haiyang Zhang, Chenxu Dai, Qingjuan Zhao, Hao feng, Zhanlin Ji and Ivan Ganchev	ESN 'DA-PN' 'DA-PN + Cover' 'DA-PN + Cover + MLO'	**29.17** **32.20** **33.43** **33.75**	**10.97** **13.46** **13.77** **14.09**	**30.95** **31.26** **32.93** **32.69**
Paper-2	Nabil Alami, Mohammed Meknassi and Noureddine En-nahnahi	Variational auto encoder model, Auto-Encoder (AE), Extreme Learning Machine Auto-Encoder model (ELM-AE)	**13.34** **14.92** **16.37**	**25.91** **28.12** **26.71**	**-** **-** **-**
Paper-3	Shuai Zhao, Fucheng You and And Zeng Yuan Liu	BSA BSA* (Encoder (BERT) and Decoder (Transformer))	**40.4** **44.2**	**25.9** **28.9**	**36.7** **39.2**
Paper-4	Fucheng You, Shuai Zhao and And Jingjing Chen	TIF-SR (On LCSTS dataset) (On NLPCC2017 Dataset)	 **42.1** **38.9**	 **28.6** **24.6**	 **37.4** **33.7**
Paper-5	Akanksha J, Alegre E, Fidalgo E and Laura Fernández-Robles	SummCoder (On AE-Net1) (On DUC 2002 Dataset) (On TIDSumm Dataset) (On Blog Summarization Dataset)	 **51.7** **51.7** **58.8** **78.0**	 **27.5** **27.5** **48.9** **71.7**	 **44.6** **44.6** **49.3** **72.7**
Paper-6	Swaranjali Jugran, Ashish Kumar, Bhupendra Singh Tyagi and Mr. Vivek Anand	SpaCy & NLTK	-	-	-

6.4 Proposed System

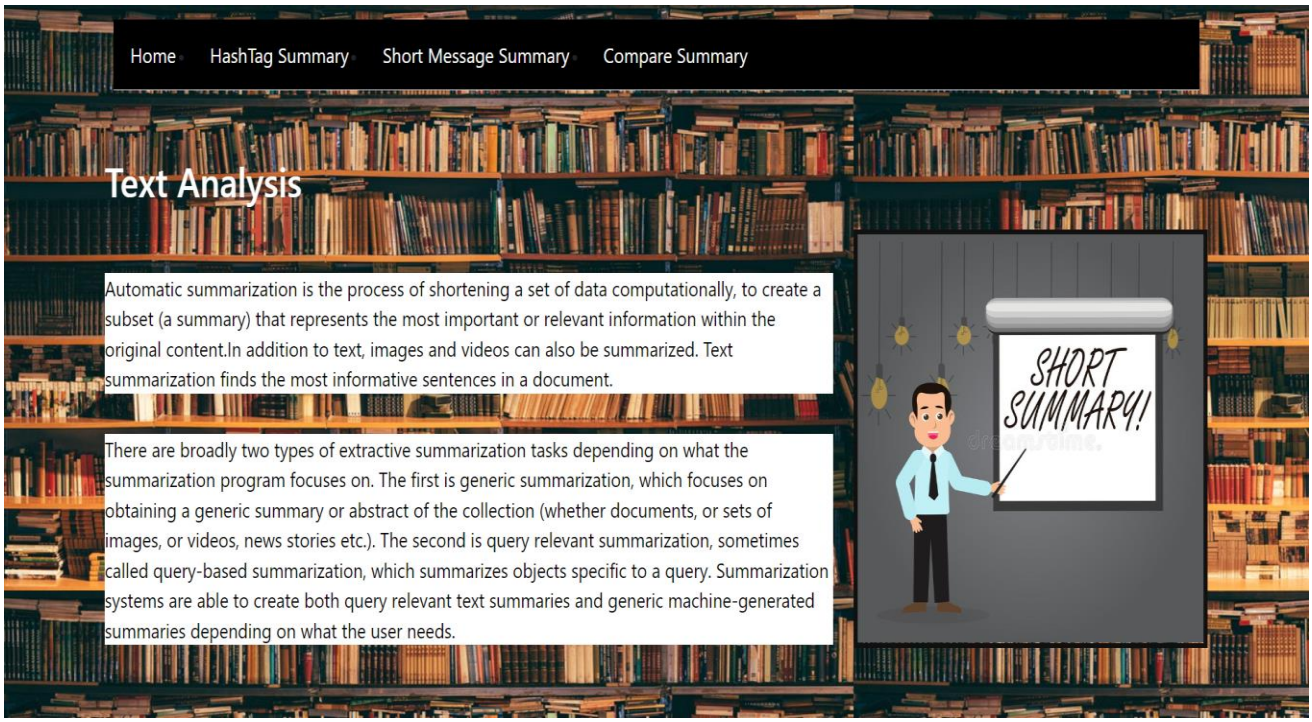

Figure 6.1: Screenshot of the Home Page

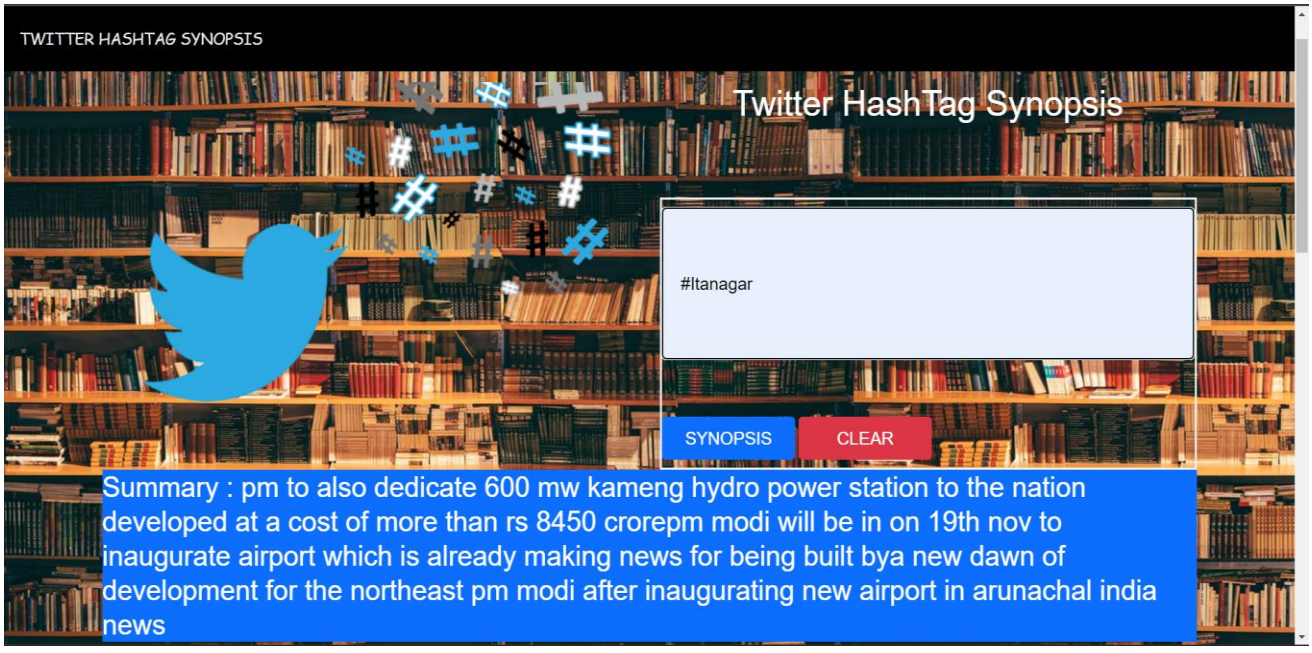

Figure 6.2: Screenshot of the Twitter Hash Tag Summary Page

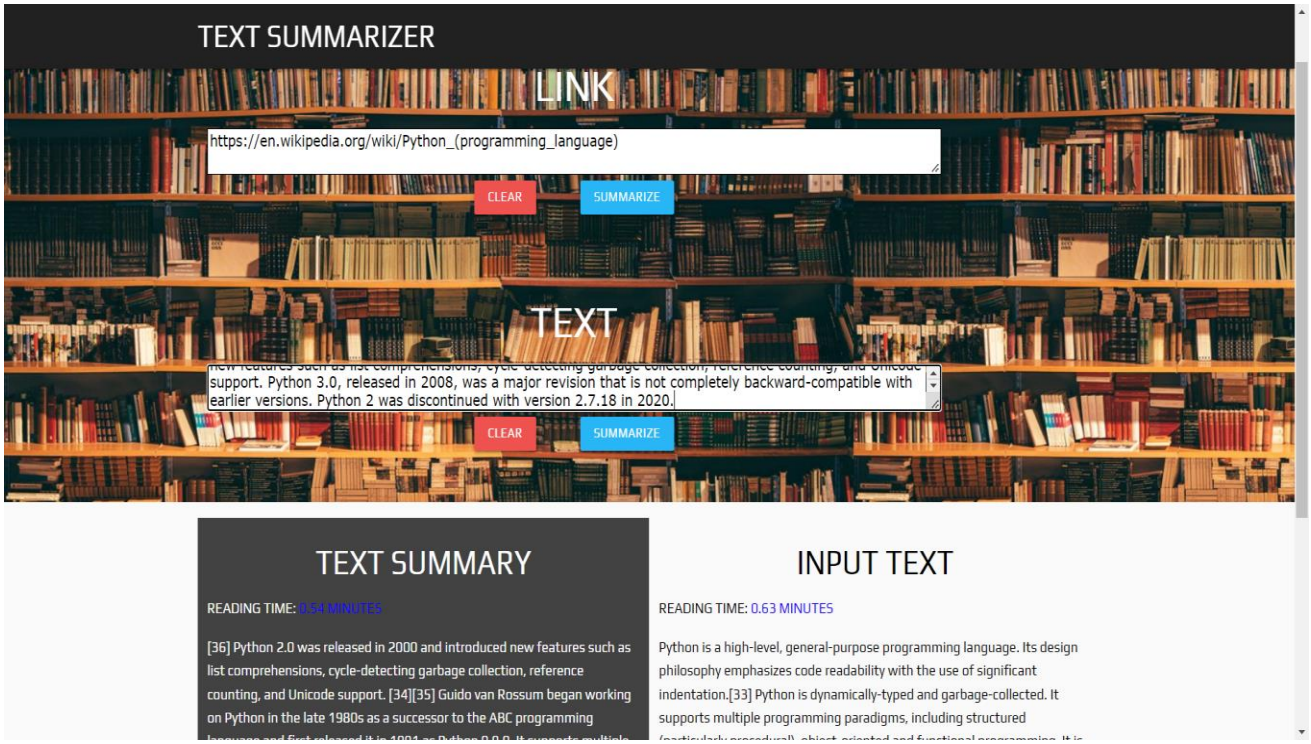

Figure 6.3: Screenshot of the Url & Text Summary Page

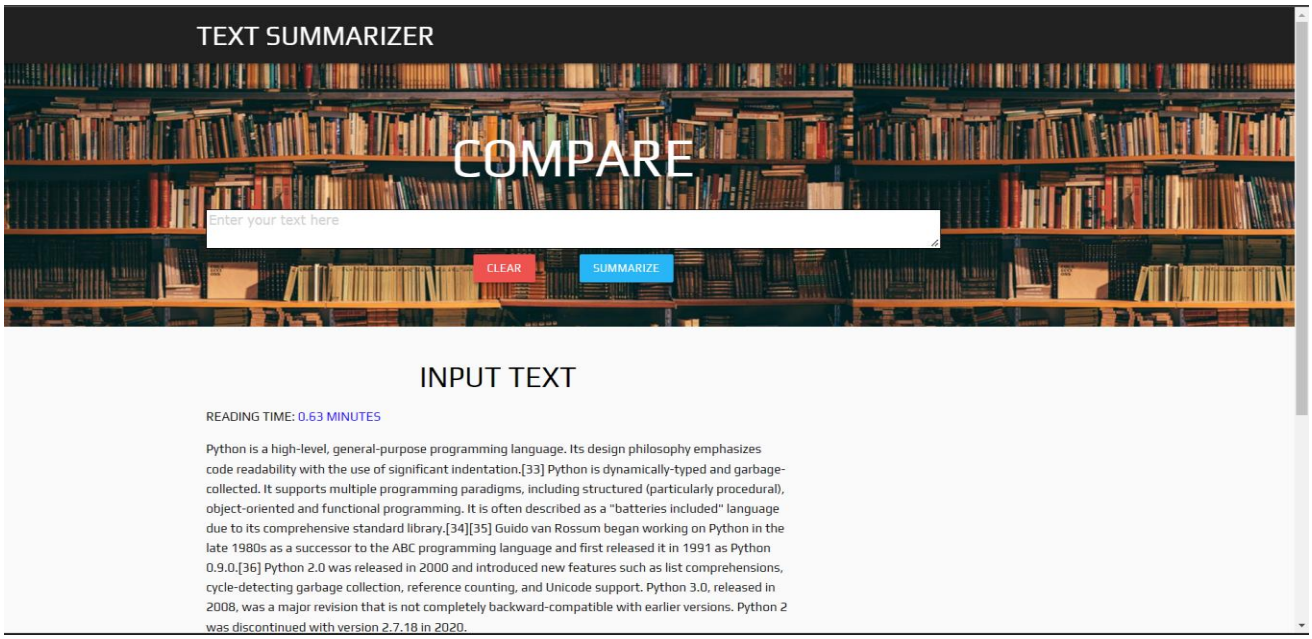

Figure 6.5: Method Comparison Page

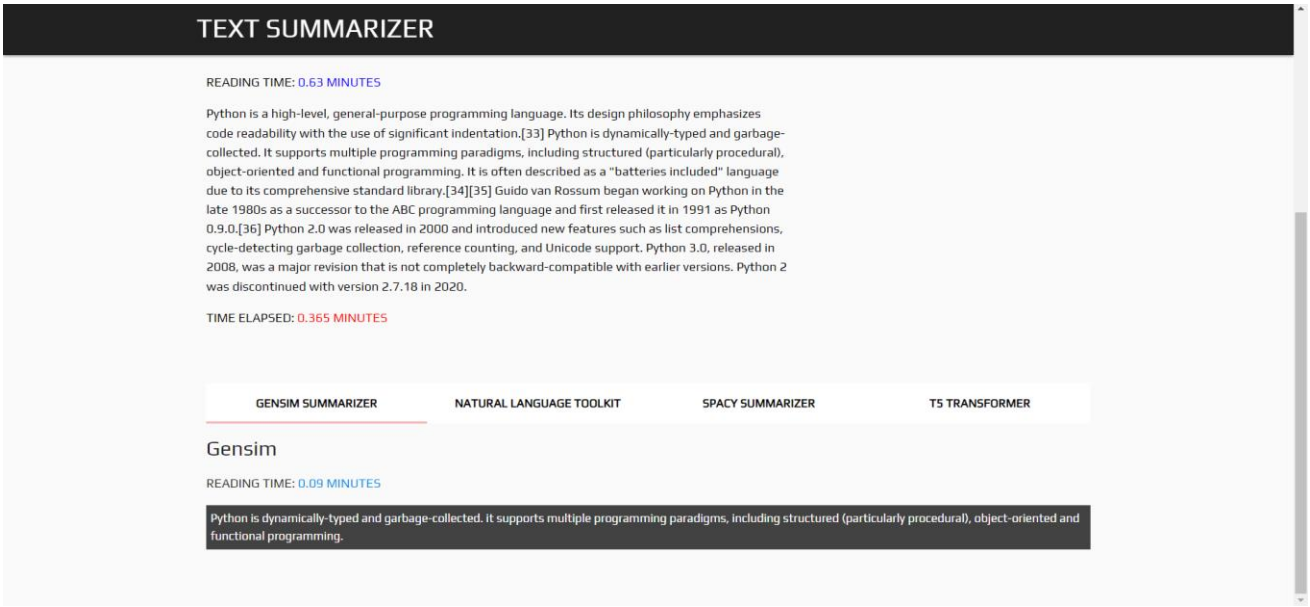

Figure 6.5: Method Comparison Page

Chapter 7

CONCLUSION AND FUTURE SCOPE

The main aim of our project is to make summary of the text and twitter tweets, Text & Tweet Summarization has been implemented using NLP approaches. NLP is a field that must always advance in keeping with contemporary technological advancements. The summarizing sector will greatly benefit from the incorporation of these techniques, both for consumers and developers. In these approaches NLTK & SpaCy are mainly used along with Text Rank algorithm. We assess this attempt as successful, especially considering the ROUGE score. Our proposed method has proven good accuracy compared with hybrid ML & DL algorithms. Finally we get a summary of tweets based on user hashtag along with manual and URL summarization. To develop more workable answers to the issues faced by many readers, additional research must be conducted in this area.

The future scope of this work will be to answer the real-time issues that are frequently experienced by users and need to be solved utilizing the advanced methodologies. In this project we have used the twitter API which is used to access the tweets from twitter which took high time. May this time can be reduced using better approaches. The accuracy of hashtag summary will depend on the tweets made on this hashtag in twitter.

Chapter 8

REFERENCES

[1] Jiang, J., Zhang, H., Dai, C., Zhao, Q., Feng, H., Ji, Z., & Ganchev, I. (2021). Enhancements of attention-based bidirectional lstm for hybrid automatic text summarization. IEEE Access, 9, 123660-123671.

[2] Alami, N., Meknassi, M., & En-nahnahi, N. (2019). Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. Expert systems with applications, 123, 195-211.

[3] Zhao, S., You, F., & Liu, Z. Y. (2020). Leveraging Pre-Trained language model for summary generation on short text. IEEE Access, 8, 228798-228803.

[4] You, F., Zhao, S., & Chen, J. (2020). A topic information fusion and semantic relevance for text summarization. IEEE Access, 8, 178946-178953.

[5] Joshi, A., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019). SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders. Expert Systems with Applications, 129, 200-215.

[6] JUGRAN, S., KUMAR, A., TYAGI, B. S., & ANAND, V. (2021, March). Extractive automatic text summarization using SpaCy in Python & NLP. In 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 582-585). IEEE.

[7] Li, Z., Peng, Z., Tang, S., Zhang, C., & Ma, H. (2020). Text summarization method based on double attention pointer network. IEEE Access, 8, 11279-11288.

[8] Adhikari, S. (2020, March). Nlp based machine learning approaches for text summarization. In 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC) (pp. 535-538). IEEE.

[9] Irani, D., Webb, S., Pu, C., & Li, K. (2010). Study of trend-stuffing on twitter through text classification. In Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS).

[10] Afsharizadeh, M., Ebrahimpour-Komleh, H., & Bagheri, A. (2018, April). Query-oriented text summarization using sentence extraction technique. In 2018 4th international conference on web research (ICWR) (pp. 128-132). IEEE.

APPENDIX A

APPENDIX A

Summarizer.py

```
import nltk
nltk.download('punkt')
from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize,sent_tokenize

import heapq
import os
def nltk_summarizer(raw_text):
  stopWords = set(stopwords.words("english"))
  word_frequencies = {}
  for word in nltk.word_tokenize(raw_text):
       if word not in stopWords:
            if word not in word_frequencies.keys():
                 word_frequencies[word] = 1
            else:
                 word_frequencies[word] += 1

  maximum_frequncy = max(word_frequencies.values())
  for word in word_frequencies.keys():
       word_frequencies[word] =
(word_frequencies[word]/maximum_frequncy)
  sentence_list = nltk.sent_tokenize(raw_text)
  sentence_scores = {}
  for sent in sentence_list:
       for word in nltk.word_tokenize(sent.lower()):
            if word in word_frequencies.keys():
                 if len(sent.split(' ')) < 30:
                      if sent not in sentence_scores.keys():
                           sentence_scores[sent] =
word_frequencies[word]
```

```
                    else:
                        sentence_scores[sent] += word_frequencies[word]
  summary_sentences = heapq.nlargest(7, sentence_scores, key=sentence_scores.get)

  summary = ' '.join(summary_sentences)
  return summary

data=open("C:\\Users\\Venkat\\twitter-trends-summarizer-master\\tweets\\cleanedcleaned_data",'r')
rawtext=""
count=0
for line in data:
    line=line.replace("\n",".")
    if len(line)>20:
        count+=1
        rawtext=rawtext+line
    if count==300:
        break
output=nltk_summarizer(rawtext)
print(output)
print(len(output))
```

Ch_6.ipynb:

```
import pandas as pd
import numpy as np
import nltk
import re
from nltk.tokenize import  sent_tokenize,word_tokenize
import networkx as nx
from sklearn.metrics.pairwise import cosine_similarity
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
##Acessing of the sentences and tokenizing the paragraph into
sentences


def tokenization():


data=open("C:\\Users\\Venkat\\AppData\\Local\\Programs\\Python\\
wt1\\cleaned.txt",'r')
  rawtext=""
  count=0
  sentences=[]
  for line in data:


      sentences.append(sent_tokenize(line))




  sentences = [y for x in sentences for y in x]


  clean_sentences=pd.Series(sentences).str.replace("[^a-zA-Z]","
")
  clean_sentences=[i.lower() for i in clean_sentences]
  return clean_sentences,sentences


def remove_stopwords(sen):
    stop_words=stopwords.words('english')
    sen_new = " ".join([i for i in sen if i not in stop_words])
    return sen_new
def textRankAlgo():
  clean_sentences,sentences=tokenization()
  clean_sentences = [remove_stopwords(r.split()) for r in
clean_sentences]
  word_embeddings = {}
  f = open('C:\\Users\\Venkat\\twitter-trends-summarizer-
```

```
master\\glove.6B.100d.txt', encoding='utf-8')
  for line in f:
      values = line.split()
      word = values[0]
      coefs = np.asarray(values[1:], dtype='float32')
      word_embeddings[word] = coefs
  f.close()

  sentence_vectors = []
  for i in clean_sentences:
    if len(i) != 0:
      v = sum([word_embeddings.get(w, np.zeros((100,))) for w in
          i.split()])/(len(i.split())+0.001)
    else:
      v = np.zeros((100,))
    sentence_vectors.append(v)

  sim_mat = np.zeros([len(sentences), len(sentences)])
  for i in range(len(sentences)):
    for j in range(len(sentences)):
      if i != j:
        sim_mat[i][j] = cosine_similarity
(sentence_vectors[i].reshape(1,100),sentence_vectors[j].reshape(
1,100))[0,0]
  nx_graph = nx.from_numpy_array(sim_mat)
  scores = nx.pagerank(nx_graph)
  ranked_sentences = sorted(((scores[i],s) for i,s in
enumerate(sentences)), reverse=True)

  return (ranked_sentences[0][1]+ranked_sentences[1][1]
```

APPENDIX B

APPENDIX B

app.py:

```
from trending_london import Acess_data
from cleaned import cleaning
from textrank_summary import textRankAlgo
from summarizer import nltk_summarizer
from textrank_dup import textRankAlgo1
from summarizer_T5 import t5transformers



from bs4 import BeautifulSoup
from urllib.request import urlopen
from flask import Flask,render_template,redirect,url_for,request
import time



app=Flask(__name__)

def readingTime(mytext):
    words=mytext.split(" ")
    total_words = len(words)
    estimatedTime = total_words/200.0
    return estimatedTime

def get_text(url):
  page = urlopen(url)
  soup = BeautifulSoup(page)
  fetched_text = ' '.join(map(lambda
p:p.text,soup.find_all('p')))
  return fetched_text
```

```
@app.route('/')
def home():
    return render_template('navigation.html')
#navigation bar
@app.route('/twitter')
def twitter():
    return render_template('twitter.html')

@app.route('/short_text')
def short_text():
    return render_template('short_text.html')



@app.route('/synopsis',methods=['GET','POST'])
def synopsis():
    if request.method=='POST':
        text=request.form["url"]
    Acess_data(text)
    filename=cleaning()
    summary=textRankAlgo()

    return render_template("twitter.html",pred=summary)


@app.route('/compare_summary')
def compare_summary():
    return render_template('compare_summary.html')

@app.route('/comparer',methods=['GET','POST'])
def comparer():
```

```
  start = time.time()
  if request.method == 'POST':
        rawtext = request.form['rawtext']
        final_reading_time = readingTime(rawtext)
        final_summary_spacy = textRankAlgo1(rawtext)
        summary_reading_time = readingTime(final_summary_spacy)
        # Gensim Summarizer
        final_summary_gensim = t5transformers(rawtext)
        summary_reading_time_gensim =
readingTime(final_summary_gensim)
        # NLTK
        final_summary_nltk = nltk_summarizer(rawtext)
        summary_reading_time_nltk =
readingTime(final_summary_nltk)
        # Sumy
        final_summary_sumy =textRankAlgo1(rawtext)
        summary_reading_time_sumy =
readingTime(final_summary_sumy)


        end = time.time()
        final_time = final_reading_time-summary_reading_time
  return
render_template('compare_summary.html',ctext=rawtext,final_summa
ry_spacy=final_summary_spacy,final_summary_gensim=final_summary_
gensim,final_summary_nltk=final_summary_nltk,final_time=final_ti
me,final_reading_time=final_reading_time,summary_reading_time=su
mmary_reading_time,summary_reading_time_gensim=summary_reading_t
ime_gensim,final_summary_sumy=final_summary_sumy,summary_reading
_time_sumy=summary_reading_time_sumy,summary_reading_time_nltk=s
ummary_reading_time_nltk)


@app.route('/analyze',methods=['GET','POST'])
def analyze():
  start = time.time()
  if request.method == 'POST':
        rawtext = request.form['rawtext']
```

```
        final_reading_time = readingTime(rawtext)
        final_summary = nltk_summarizer(rawtext)
        summary_reading_time = readingTime(final_summary)
        end = time.time()
        final_time = end-start
    return
render_template('short_text.html',ctext=rawtext,final_summary=fi
nal_summary,final_time=final_time,final_reading_time=final_readi
ng_time,summary_reading_time=summary_reading_time)


@app.route('/analyze_url',methods=['GET','POST'])
def analyze_url():
    start = time.time()
    if request.method == 'POST':
        raw_url = request.form['raw_url']
        rawtext = get_text(raw_url)
        final_reading_time = readingTime(rawtext)
        final_summary =nltk_summarizer(rawtext)
        summary_reading_time = readingTime(final_summary)
        end = time.time()
        final_time = end-start
    return
render_template('short_text.html',ctext=rawtext,final_summary=fi
nal_summary,final_time=final_time,final_reading_time=final_readi
ng_time,summary_reading_time=summary_reading_time)


if __name__=="__main__":
    app.run(debug=True)
```