

Vivesvaraya Technological University

"Jnana Sangama", Belagavi - 590018



**A Project Report on
Text Summarizer using NLP**

*Submitted in partial fulfilment of the requirement
for the award of the degree of*

**BACHELOR OF ENGINEERING
IN INFORMATION SCIENCE AND
ENGINEERING**

Submitted by

Nandini Mundra	4NI16IS055
Sushant Kr Srivastava	4NI16IS109
Tanmoy Debnath	4NI16IS111
Vinay Pandhariwal	4NI16IS116

Under the Guidance of

Ms .Ranjitha K S B.E, M.Tech
Department of ISE, NIE,Mysore



The National Institute of Engineering
(Autonomous Institute)
- 570008.



**Department of Information
Science and Engineering** Mysore
NIE, Mysore - 570008.

**Academic Year
2019-20**

The National Institute of Engineering
Manandavadi Road, Mysuru - 570008
Department of Information Science and Engineering



Certificate

Certified that the project work entitled "**Text Summarizer using NLP**" is the bona fide work carried out by

Nandini Mundra

4NI16IS055

Sushant Kr Srivastava

4NI16IS109

Tanmoy Debnath

4NI16IS111

Vinay Pandhariwal

4NI16IS116

in partial fulfilment for the requirements of the eight semester BE in Information Science & Engineering prescribed by The National Institute of Engineering, Autonomous Institution under Visvesvaraya Technological University, Belagavi. It is certified that all corrections or suggestions indicated for Internal Assessment have been incorporated. The Project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the eighth semester.

Signature of Guide
Ms. Ranjitha K S

Signature of HoD
Dr. P. Devaki

Signature of Principal
Dr. Rohini Nagapadma

Name of the Examiner

Signature with Date

1

2

ACKNOWLEDGEMENT

We would like to take this opportunity to express our profound gratitude to all those people who are directly or indirectly involved in the completion of the project. We thank each and every one who encouraged us in every possible way.

We would like to thank **Dr.Rohini Nagapadma**, principal, NIE, Mysuru for letting us to be a part of this prestigious institution and letting us to explore our abilities to the fullest.

We would like to extend our sincere gratitude to **Dr. P. DEVAKI**, Professor and HOD, Department of ISE, NIE, Mysuru for being a source of inspiration and instilling an enthusiastic spirit in us throughout the process of project making.

We wish to express our heartfelt gratitude towards **Ms.Ranjitha K S**, Minor project Guide, Assistant Professor, Department of ISE, NIE, Mysuru for his consistent guidance and valuable knowledge.

We are extremely pleased to thank our family members and friends for their continuous support, inspiration and encouragement, for their helping hand and also last but not the least We are grateful to all the members who supported us directly or indirectly in our academic process.

NANDINI MUNDRA: 4NI16IS055

SUSHANT SRIVASTAVA: 4NI16IS109

TANMOY DEBNATH: 4NI16IS111

VINAY PANDHARIWAL: 4NI16IS116

Abstract

In today's world who has time to look over the entire article, blog, research papers. So, automatic text summarization can be a big help in these use cases. Taking inspiration from applications like Inshorts, who have leveraged this technology for long, this technology can be used for technical content sources such as geeksforgeeks, stackoverflow, stackexchange and various other web sources. This system can help in getting a reliable, fast and concise information on IT buzzwords, core topics used in IT industry.

Text summarization is the process of generating short, fluent, and most importantly accurate summary of a respectively longer text document . The main idea behind automatic text summarization is to be able to find a short subset of the most essential information from the entire set and present it in a human-readable format. As online textual data grows, automatic text summarization methods have potential to be very helpful because more useful information can be read in a short time.

A new feature has also been added to make people get rid of seeing the monotonous code and just using the UI to understand the functionality of the test summarizer and get the results.

TABLE OF CONTENTS

SI No.	Topic	Page No.
1	Introduction	4
	1.1 Project purpose	4
	1.2 Existing system	5
	1.3 Proposed system	5
2	Literature Survey	7
3	System Requirements	9
4	System Design	10
	4.1 Reading from CSV file	10
	4.2 Pre-processing	10
	4.2.1 Introduction	10
	4.2.2 Implementation	11
	4.2.2.1 Tokenization	12
	4.2.2.2 Stop word removal	13
	4.2.2.3 Deduplication	14
	4.3 Lex Algorithm	14
	4.4 Genism Algorithm	16
5	System Implementation	17
6	Testing	19
	6.1 Unit testing	19
	6.2 Integration testing	19
	6.3 Validation testing	19
	6.4 Table representing multiple tests	20
7	Deployment	23
	Conclusion	25
	Future Enhancement	26
	References	27

LIST OF FIGURES

1.1 Steps of Project.....	6
4.1 Pre-processing steps.....	10
5.1 code for tokenizing the text and then using NLTK libraries to..... print summary	17
5.2 code for printing word count and then printing their respective..... summary	17
5.3 Output asking for the options to feed input.....	18
5.4 Final Output.....	18
7.1 The user interface.....	23
7.2 User interface with text added.....	23
7.3 code part 1.....	24
7.4 code part 2.....	24

Chapter 1

INTRODUCTION

1.1 Project Purpose

In today's world the amount of data present online it's very difficult to summarize the entire content and refine it into suitable form of information. The abundance of unstructured information increases the need for automatic systems that can “condense” information from various documents into a shorter- length, readable summary. Such summaries may be further required to cover a specific information needed (e.g., summarizing web search results, medical records, question answering and more).

In our system, we will take data from various sources for a particular topic and summarize it for the convenience of the people, so that they don't have to go through so multiple sites for relevant data. In order to solve the problem of visiting N sites and then getting relevant information we have a solution that is Automatic text summarization, which helps us summarize the entire context and gives you a brief summary based on what is relevant. Various methods have been proposed for this summarization task. These methods can be categorized based on two main dimensions i.e. extractive versus abstractive and supervised versus unsupervised. What we are going to use is the Extractive and Unsupervised learning technique. Extractive methods generate a summary using only text fragments extracted from the document(s) and Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance . Here the task of the machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Automatic text summarization is a machine learning technique that shortens the text document to create summary for our convenience. Now basically what we are using here is NLP i.e. Natural Language Processing. Summarization technologies typically include the following functionalities: identification of important information in source documents, identification and removal of duplicate information, summary information coverage optimization, and content ordering and rewriting to enhance readability

1.2 Existing System

The existing system for technical and technological concepts doesn't exist. There are very few applications such as Inshorts which is a news based application. Inshorts takes news articles from sources like Hindustan Times, The Hindu etc. and perform extractive text summarization to get 60 word summarized news articles which is really helpful in this busy world.

Some technical websites such as Geeksforgeeks.org and tutorialspoint.com provide summaries of technical concepts but there are manually done which takes expertise and a lot of hard work and research to deliver reliable and concise summaries. These websites don't cover buzzwords like Big data analytics, Cryptography, data mining. Looking at these limitations, automatic text summarization can be really helpful for students, researchers, teachers, lecturers who need a quick overview of a certain technical concept. The current system provides us various pages which we can browse and gather points on our own for our reference but provides no summary of it.

1.3 Proposed System

There is nothing much that currently exists on this topic, so our project is one of its kind. Our project focuses on providing a system design which could prepare a bullet-point summary for the user by scanning through multiple articles. What we have in existing system is that they don't provide any summary mechanism that will help in getting concise and reliable summaries of technological topics such as ML, AI etc. If a reader wants to get an overview, he has to go through multiple sources such as blogs, articles, question forums etc which is very rigorous and time consuming. So we are improvising this existing system to come up with summary for the user. We will apply the Text Rank algorithm on a dataset of scraped articles with the aim of creating a nice and concise summary.

The first step would be to concatenate all the text contained in the articles. Then split the text into individual sentences. Then do pre-processing on them. In Preprocessing, we perform various operations like Stemming, Lemmatization, stopwords removal. In the next step, we will find vector representation (word embeddings) for each and every sentence. For this we have used Word2Vec model. Similarities between sentence vectors are then calculated and stored in a matrix. The similarity matrix is then converted into a graph, with sentences as

vertices and similarity scores as edges, for sentence rank calculation. For sentence ranking, text rank algorithm is used. The steps have been described in Fig. 4.1 accordingly.

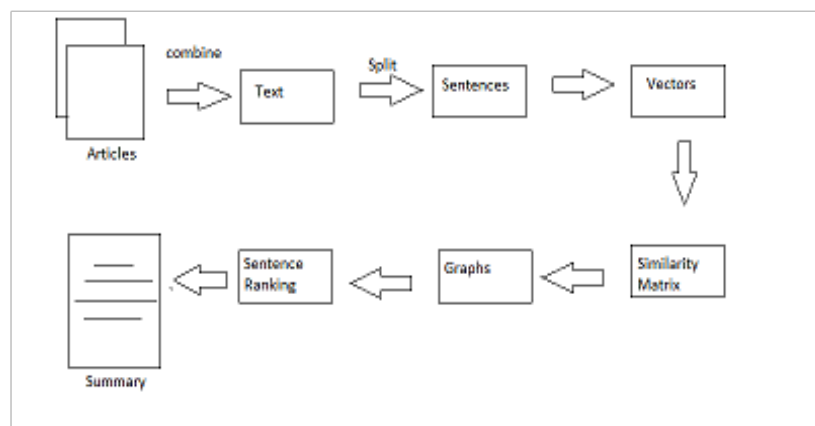


Fig. 1.1 Steps of Project

Along with this a user interface has also been created to make it feasible for people to use it.

Chapter 2

LITERATURE SURVEY

[1] **Title :** “Learning Python the Hard Way”

Author(s) : Zed Shaw

Learn Python The Hard Way takes you from absolute zero to be able to read and write basic Python to then understand other books on Python. No experience necessary to begin. If you’ve always wanted to code but have no idea where to begin, then this book is for you. You will learn Python by working through 52 brilliantly crafted exercises. Read them. Type their code precisely. Fix your mistakes. Watch the programs run. As you do, you’ll learn how software works; what good programs look like; how to read, write, and think about code; and how to find and fix your mistakes using tricks professional programmers use.

[2] **Title:** “Performing literature review using text mining, Part III: Summarizing articles”

Author(s): Dazhi Yang_ and Allan N. Zhang

Source(s): IEEE Xplore Digital Library

Text Rank, is used to perform automatic summary of academic articles, for effective and efficient literature review. The algorithm utilizes a weighted undirected graph to represent the sentences in an article, and uses Google Page Rank to order the sentences.

[3] **Title:** “Overview of extractive text summarization”

Author(s): Ali Toofanzadeh Mozhdehi, Mohamad Abdolahi and Shohreh Rad Rahimi

Source(s): IEEE Xplore Digital Library

Text Summarization is the process of creating a condensed form of text document which maintains significant information and general meaning of source text. Automatic text summarization becomes an important way of finding relevant information precisely in large text in a short time with little efforts. Text summarization approaches are classified into two categories: extractive and abstractive. This paper presents the comprehensive survey of both the approaches in text summarization.

[4] **Title:** “Text Rank algorithm by exploiting Wikipedia for short text keywords extraction”

Author(s): Wengen Li and Jiabao Zhao

Source(s): IEEE Xplore Digital Library

The characteristic of poor information of short text often makes the effect of traditional keywords extraction not as good as expected. In this paper, we propose a graph-based ranking algorithm by exploiting Wikipedia as an external knowledge base for short text keywords extraction. To overcome the shortcoming of poor information of short text, we introduce the Wikipedia to enrich the short text. We regard each entry of Wikipedia as a concept, therefore the semantic information of each word can be represented by the distribution of Wikipedia's concept. And we measure the similarity between words by constructing the concept vector. Finally we construct keywords matrix and use TextRank for keywords extraction. The comparative experiments with traditional TextRank and baseline algorithm show that our method gets better precision, recall and F-measure value. It is shown that TextRank by exploiting Wikipedia is more suitable for short text keywords extraction.

[5] Geeks for Geeks , w3 school,Vidhya Analytics.

Platforms on the Internet where one could learn about computer science and related topics from scratch.

Chapter 3

SYSTEM REQUIREMENT

Software Requirements

- Operating System - Windows XP, Windows 10, MAC OS
- Front End tool – Pycharm
- Back End tool – Text Datasets

Hardware Requirements

- System – Intel i5, 2.4 GHz
- Hard disk - 40Gb
- RAM – 1Gb

CHAPTER 4

SYSTEM DESIGN

4.1 READING FROM CSV FILE

We read the data from our CSV file which has our datasets and then concatenate them and form a text, now we split the text to individual sentences and do the further step on top of that. The columns are article_id, article_text, source,topic and when we do other tasks all of them is going to be applied on to article_text.

4.2 PREPROCESSING

4.2.1 Introduction

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. The following Fig shows the Pre-Processing Steps.

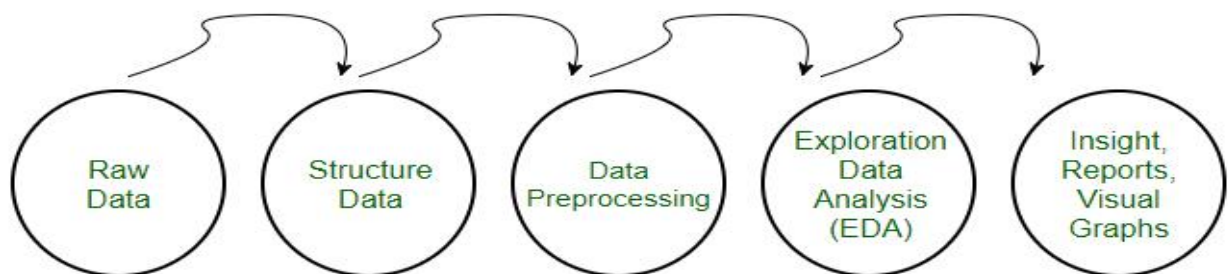


Fig. 4.1 Pre-Processing Steps

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set. Another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and best out of them is chosen.

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing.

Data goes through a series of steps during pre-processing:

- **Data Cleaning:** Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.
- **Data Integration:** Data with different representations are put together and conflicts within the data are resolved.
- **Data Transformation:** Data is normalized, aggregated and generalized.
- **Data Reduction:** This step aims to present a reduced representation of the data in a data warehouse.
- **Data Discretization:** Involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals.
- Data preprocessing is used database-driven applications such as customer relationship management and rule based applications(like neural networks).

4.2.2 Implementation

Steps for pre-processing.

- Tokenization
- Stop word removal
- Deduplication

We have used NLTK for this. NLTK is one of the leading platforms for working with human language data and Python, the module NLTK is used for natural language processing. NLTK is literally an acronym for Natural Language Toolkit. The NLTK module is a massive tool kit, aimed at helping you with the entire Natural Language Processing (NLP) methodology. NLTK will aid you with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, and then even

with helping your machine to understand what the text is all about. In this series, we're going to tackle the field of opinion mining, or sentiment analysis.

4.2.2.1 Tokenization

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded. The tokens become the input for another process like parsing and text mining.

Tokenization is used in computer science, where it plays a large part in the process of lexical analysis.

Tokenization relies mostly on simple heuristics in order to separate tokens by following a few steps:

- Tokens or words are separated by whitespace, punctuation marks or line breaks
- White space or punctuation marks may or may not be included depending on the need
- All characters within contiguous strings are part of the token. Tokens can be made up of all alpha characters, alphanumeric characters or numeric characters only.

Tokens themselves can also be separators. For example, in most programming languages, identifiers can be placed together with arithmetic operators without white spaces. Although it seems that this would appear as a single word or token, the grammar of the language actually considers the mathematical operator (a token) as a separator, so even when multiple tokens are bunched up together, they can still be separated via the mathematical operator.

Each "entity" that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.

EXAMPLE_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great, and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'Python', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', 'n't', 'eat', 'cardboard', '.']
```

4.2.2.2 Stop Word Removal

The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the `nltk_data` directory. *home/pratima/nltk_data/corpora/stopwords* is the directory address.(Do not forget to change your home directory name)

Text may contain stop words like ‘the’, ‘is’, ‘are’. Stop words can be filtered from the text to be processed. There is no universal list of stop words in nlp research, however the `nltk` module contains a list of stop words.

When computers process natural language, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words.

For example, if you give the input sentence as –

John is a person who takes care of the people around him.

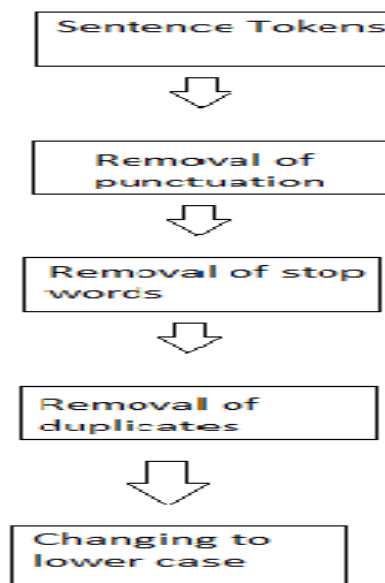
After stop word removal, you'll get the output –

```
['John', 'person', 'takes', 'care', 'people', 'around', '.']
```


4.2.2.3 Deduplication

Data deduplication is a technique for eliminating duplicate copies of repeating data. A related and somewhat synonymous term is single instance storage. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced.

Document Pre-Processing Flowchart



4.3 Lex algorithm

LexRank method for text summarization is another child method to **PageRank** method with a sibling TextRank. It uses a graph based approach for automatic text summarization. In this article we will try to learn the concept of LexRank and various methods to implement the same in Python.

LexRank is an unsupervised graph based approach for automatic text summarization. The scoring of sentences is done using the graph method. LexRank is used for computing

sentence importance based on the concept of eigenvector centrality in a graph representation of sentences.

In this model, we have a **connectivity matrix** based on intra-sentence cosine similarity which is used as the adjacency matrix of the graph representation of sentences. This sentence extraction majorly revolves around the set of sentences with same intent i.e.

a *centroid* sentence is selected which works as the mean for all other sentences in the document. Then the sentences are ranked according to their similarities.

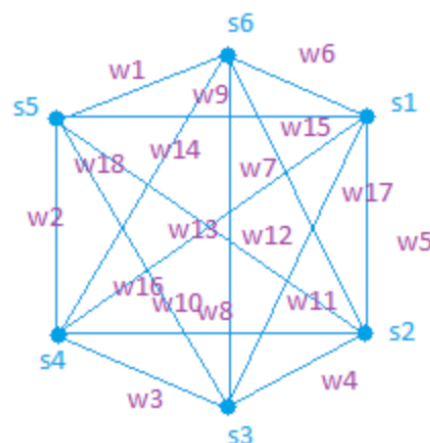
Graphical Approach

1. Based on Eigen Vector Centrality.
2. Sentences are placed at the vertexes of the Graphs
3. The weight on the Edges are calculated using cosine similarity metric.

After we construct the whole graph, we can assign weights for web pages by the following formula.

$$S(V_i) = (1 - d) + d * \sum_{j \in In(v_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

- $S(V_i)$ - the weight of webpage i
- d - damping factor, in case of no outgoing links
- $In(V_i)$ - inbound links of i, which is a set
- $Out(V_j)$ - outgoing links of j, which is a set
- $|Out(V_j)|$ - the number of outbound links



Representation of a graph

4.4 Gensim algorithm

Gensim is a free Python library designed to automatically extract semantic topics from documents.

The gensim implementation is based on the popular *TextRank* algorithm.

It is an open-source vector space modelling and topic modelling toolkit, implemented in the Python programming language, using **NumPy**, **SciPy** and optionally **Cython** for performance.

4.4.1 Text Summarisation with Gensim (TextRank algorithm)-

We use the `summarization.summarizer` from gensim.

This summarising is based on ranks of text sentences using a variation of the TextRank algorithm.

TextRank is a general purpose, **graph based** ranking algorithm for NLP.

TextRank is an automatic summarisation technique.

Graph-based ranking algorithms are a way for deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph.

Not just the algorithm , a user interface has also been created which allows used to feed their data in a text box and then select the type of algorithm they want to select.

Chapter 5

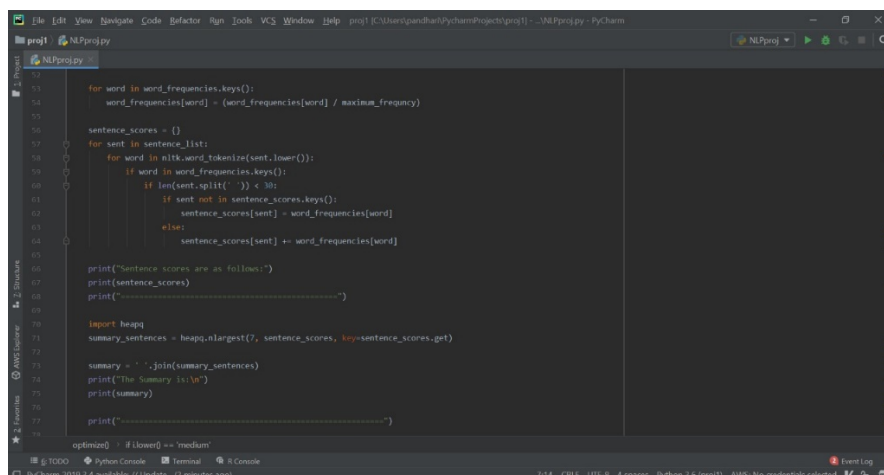
SYSTEM IMPLEMENTATION

There are multiple ways by which the input for the text summarizer can be given. They can be listed as follows:

- From any text file
- Audio input
- Input by providing URL etc..

In implementing the given system we have given all these features of feeding the input text which will be summarized and given to the user

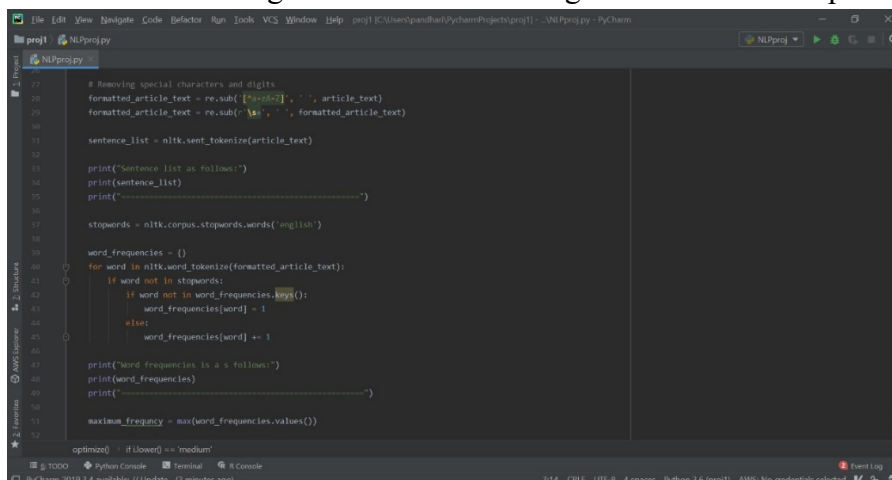
Following are the screenshots showing the code and the way in which the inputs are being given.



```

13 for word in word_frequencies.keys():
14     word_frequencies[word] = (word_frequencies[word] / maximum_frequency)
15
16 sentence_scores = {}
17 for sent in sentence_list:
18     for word in nltk.word_tokenize(sent.lower()):
19         if word in word_frequencies.keys():
20             if len(sent.split(' ')) > 30:
21                 if sent not in sentence_scores.keys():
22                     sentence_scores[sent] = word_frequencies[word]
23             else:
24                 sentence_scores[sent] += word_frequencies[word]
25
26 print("Sentence scores are as follows:")
27 print(sentence_scores)
28 print("-----")
29
30 import heapq
31 summary_sentences = heapq.nlargest(7, sentence_scores, key=sentence_scores.get)
32
33 summary = ' '.join(summary_sentences)
34 print("The summary is:\n")
35 print(summary)
36 print("-----")
37
38 optimized = if_lower() == 'medium'
  
```

Fig 5.1 code for tokenizing the text and then using NLTK libraries to print summary



```

27 # removing special characters and digits
28 formatted_article_text = re.sub('[0-9]+', '', article_text)
29 formatted_article_text = re.sub('[^a-zA-Z]', ' ', formatted_article_text)
30
31 sentence_list = nltk.sent_tokenize(article_text)
32
33 print("Sentence list as follows:")
34 print(sentence_list)
35 print("-----")
36
37 stopwords = nltk.corpus.stopwords.words('english')
38
39 word_frequencies = {}
40 for word in nltk.word_tokenize(formatted_article_text):
41     if word not in stopwords:
42         if word not in word_frequencies.keys():
43             word_frequencies[word] = 1
44         else:
45             word_frequencies[word] += 1
46
47 print("word frequencies is as follows:")
48 print(word_frequencies)
49 print("-----")
50
51 maximum_frequency = max(word_frequencies.values())
  
```

Fig 5.2 code for printing word count and then printing their respective summary

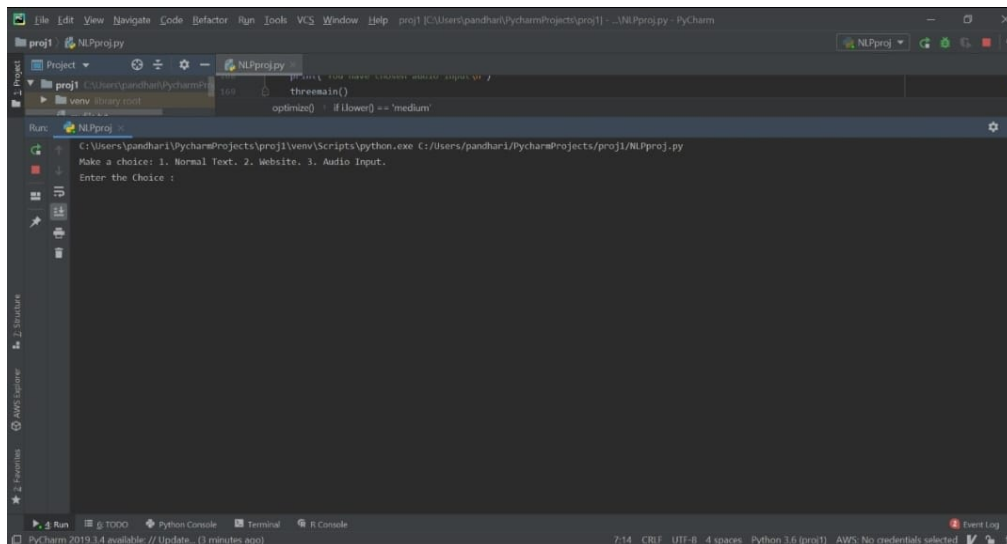


Fig 5.3 Output asking for the options to feed input

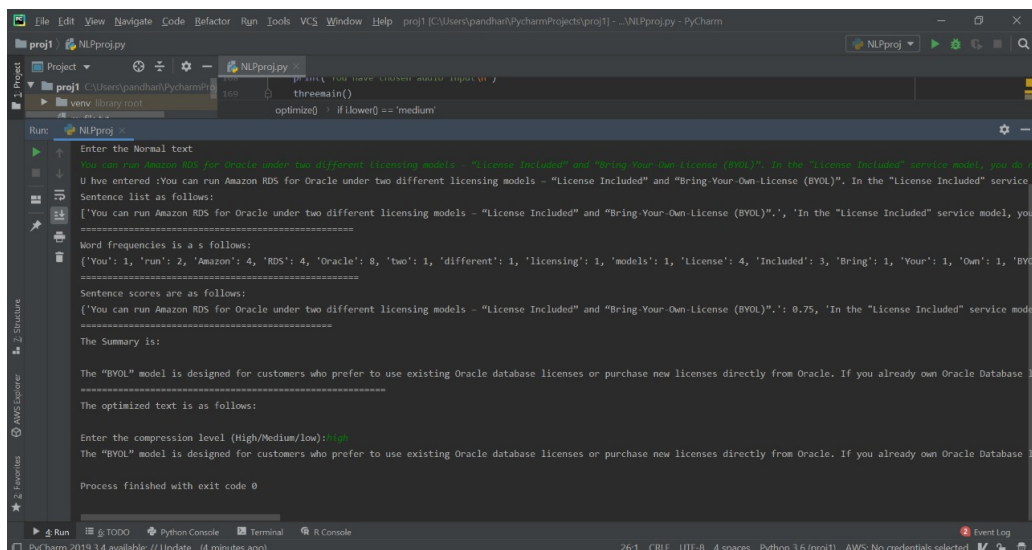


Fig 5.4 Final output

These screenshots completely show the implementation and the code which can be used by user for summarizing the text.

Chapter 6

TESTING

Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. It is the process of assessing whether an implementation confirms of its specifications. The testing phase involves the testing of the developed system using various test data.

Testing process proceeds in stages where testing is carried out incrementally in conjunction with system implementation. The Various software-testing strategies are:

1. Unit Testing
2. Integration Testing
3. Validation or Acceptance Testing

6.1 UNIT TESTING

Unit testing is performed to verify the smallest unit of each module in the system with all possible inputs.

6.2 INTEGRATION TESTING

In this testing all parts of the unit-tested functions are integrated and the entire system is tested as a whole.

6.3 VALIDATION TESTING

This tests errors and omissions. It is tested in accordance with the requirements specification and also considering other documents provided in connection to this system development.

6.4 TABLE REPRESENTING TEST CASES

Serial No.	Test Cases	Expected Result	Actual Result	Result Status
1.	Run the program and tokenize a given set of statements. For eg: Apple is a apple and is red in color.	Apple, is, a, apple, and, is, red, in, color.	Apple, is, a, apple, and, is, red, in, color.	Verified Successfully.
2.	Stop-Word Removal. For eg: Apple is a apple and is red in color.	Apple,apple, red, color.	Apple,apple, red, color.	Verified Successfully.
3.	Deduplication of dataset For eg: Apple is a apple and is red in color.	Apple, red, color.	Apple,red, color.	Verified Successfully.
4.	Validation testing: Topic not present in the dataset	Stop execution of program and notify user that topic not present in dataset	Stop execution of program and notify user that topic not present in dataset	Verified Successfully
5.	Word Vectors: If word vectors could not formed due to internal error such as API not available.	Stop program execution	Stop program execution	Verified Successfully

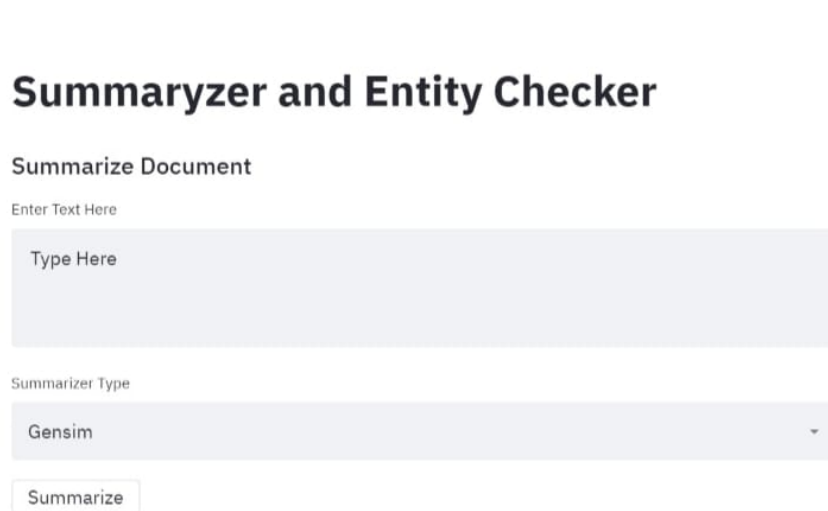
6.	Word Vectors: Vector for word in dataset not found.	Ignore the word embedding and assign 0 as word vector	Ignore the word embedding and assign 0 as word vector	Verified Successfully
7.	API not available.	Stop program execution	Stop program execution	Verified Successfully
8.	Vector formation of data not done due to internal error	Stop program execution and notify user.	Stop program execution and notify user.	Verified Successfully
9.	Graph Formation: Based on distance between sentence vectors form clusters.	Form Graphs and find most relevant edge in the graph	Form graphs and find most relevant graph	Verified Successfully
10.	Similarity matrix: Dimensions of matrix not same	Ignore the dimension and proceed further	Ignore the dimension and proceed further	Verified Successfully
11.	Similarity Matrix: Distance between sentence vectors are zero.	Assign score to be zero for the corresponding sentence vector pair.	Assign score to be zero for the corresponding sentence vector pair.	Verified Successfully
12.	Graph representation: Similarity matrix	Reshape the matrix to common	Reshape the matrix to common	Verified successfully

13.	dimensions are not same Page Rank: Page Rank API not available	dimension Stop the execution of program.	dimension Stop the execution of program.	Verified Successfully
14.	Validation testing: Dataset containing special or invalid characters	Process and remove these characters from dataset in preprocessing stage.	Process and remove these characters from dataset in preprocessing stage.	Verified successfully

Chapter 7

DEPLOYMENT

The whole system when handed over to the customers or the end users looks like fig 7.1 . This is how the user will deal with the summarizer and will select any algorithm out of the 2 given choices.



The screenshot shows a web application titled "Summaryzer and Entity Checker". Below the title is a section "Summarize Document". It contains a text input area labeled "Enter Text Here" with a placeholder "Type Here". Below the input area is a dropdown menu labeled "Summarizer Type" with "Gensim" selected. At the bottom of the form is a "Summarize" button.

Fig. 7.1 The user interface

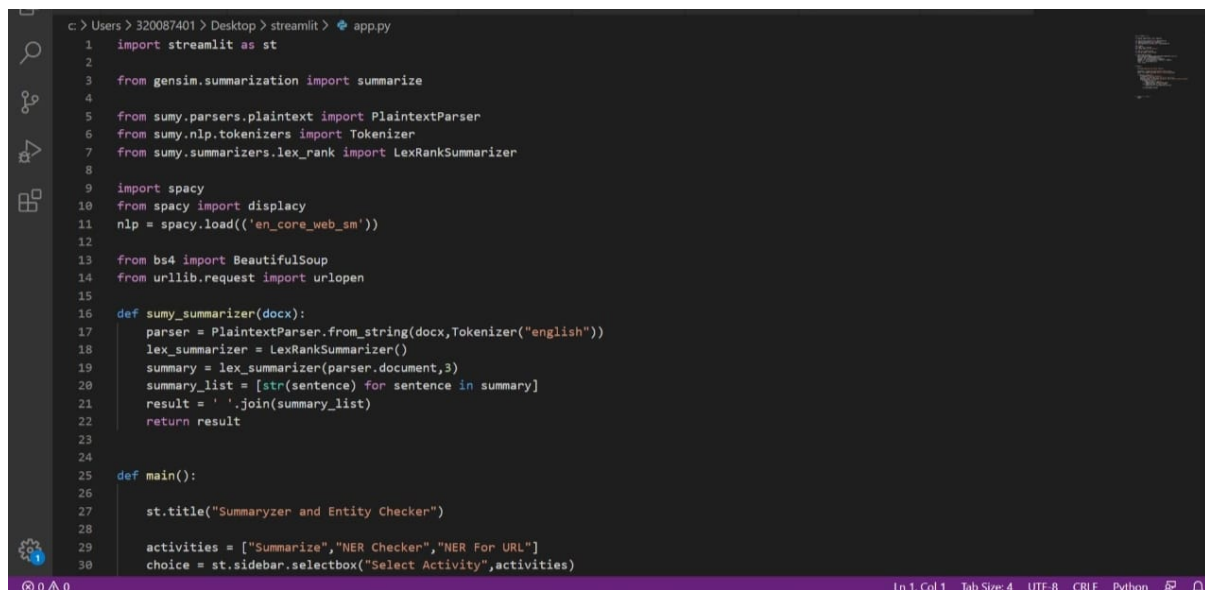


The screenshot shows the same web application as Fig. 7.1, but with text entered in the input field. The text is: "Warren Buffett founded The Giving Pledge, whereby they and other billionaires pledge to give at least half of their wealth to philanthropy.[17] The foundation says that it works to save lives and improve global health, and is working with Rotary International to eliminate polio.[18]". The "Summarizer Type" dropdown is still set to "Gensim", and the "Summarize" button is visible. Below the input field, there is a preview of the summarized text: "He is best known as the co-founder of Microsoft Corporation.[2] During his career at Microsoft, Gates held the positions of chairman, chief executive officer (CEO), president and chief software architect, while also being the largest individual shareholder until May 2014."

Fig 7.2 User interface with text added

This is the code which has been used for the design and development of the front end tool.

This has the functionality of selecting either Sumi Lex rank or Genism algorithm and then do the specific functionality related to them.

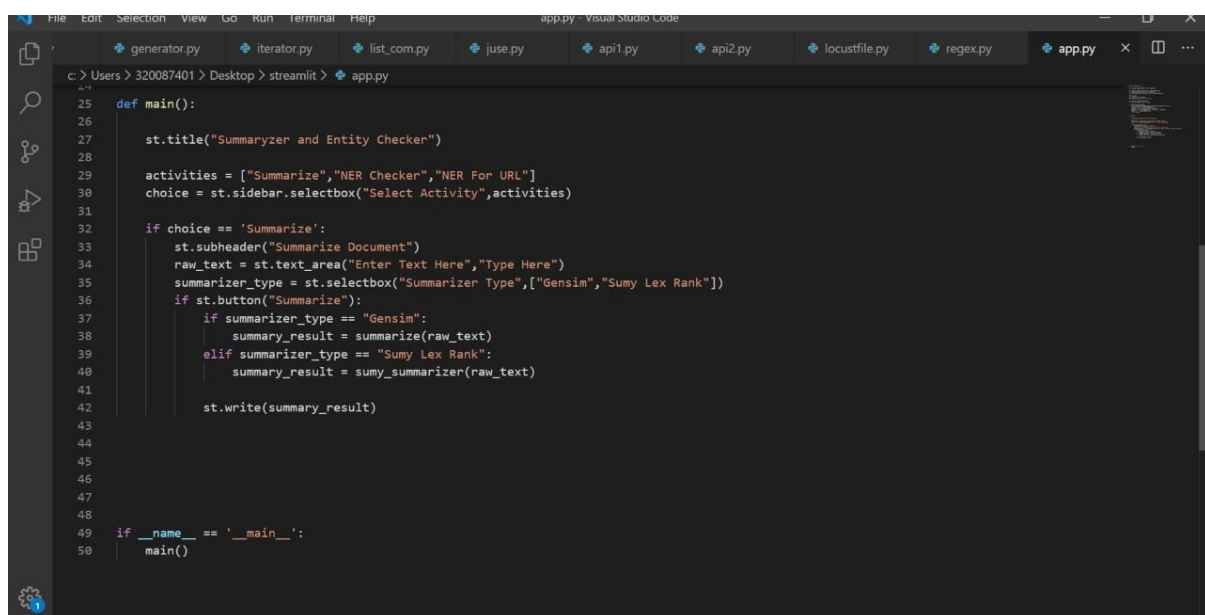


```

c> Users > 320087401 > Desktop > streamlit > app.py
1  import streamlit as st
2
3  from gensim.summarization import summarize
4
5  from sumy.parsers.plaintext import PlaintextParser
6  from sumy.nlp.tokenizers import Tokenizer
7  from sumy.summarizers.lex_rank import LexRankSummarizer
8
9  import spacy
10 from spacy import displacy
11 nlp = spacy.load(('en_core_web_sm'))
12
13 from bs4 import BeautifulSoup
14 from urllib.request import urlopen
15
16 def sumy_summarizer(docx):
17     parser = PlaintextParser.from_string(docx,Tokenizer("english"))
18     lex_summarizer = LexRankSummarizer()
19     summary = lex_summarizer(parser.document,3)
20     summary_list = [str(sentence) for sentence in summary]
21     result = ' '.join(summary_list)
22     return result
23
24
25 def main():
26
27     st.title("Summaryzer and Entity Checker")
28
29     activities = ["Summarize","NER Checker","NER For URL"]
30     choice = st.sidebar.selectbox("Select Activity",activities)

```

Fig 7.3 code part 1



```

File Edit Selection View Go Run Terminal Help
app.py - Visual Studio Code
generator.py iterator.py list_com.py juse.py api1.py api2.py locustfile.py regex.py app.py x ...
c> Users > 320087401 > Desktop > streamlit > app.py
25 def main():
26
27     st.title("Summaryzer and Entity Checker")
28
29     activities = ["Summarize","NER Checker","NER For URL"]
30     choice = st.sidebar.selectbox("Select Activity",activities)
31
32     if choice == 'Summarize':
33         st.subheader("Summarize Document")
34         raw_text = st.text_area("Enter Text Here","Type Here")
35         summarizer_type = st.selectbox("Summarizer Type",["Gensim","Sumy Lex Rank"])
36         if st.button("Summarize"):
37             if summarizer_type == "Gensim":
38                 summary_result = summarize(raw_text)
39             elif summarizer_type == "Sumy Lex Rank":
40                 summary_result = sumy_summarizer(raw_text)
41
42             st.write(summary_result)
43
44
45
46
47
48
49 if __name__ == '__main__':
50     main()

```

Fig 7.4 code part 2

CONCLUSION

Automatic text summarization is an old challenge but the current research direction diverts towards emerging trends in biomedicine, product review, education domains, emails and blogs. This is due to the fact that there is information overload in these areas, especially on the World Wide Web. Automated summarization is an important area in NLP (Natural Language Processing) research. It consists of automatically creating a summary of one or more texts. The purpose of extractive document summarization is to automatically select a number of indicative sentences, passages, or paragraphs from the original document. Text summarization approaches based on Neural Network, Graph Theoretic, Fuzzy and Cluster have, to an extent, succeeded in making an effective summary of a document. Both extractive and abstractive methods have been researched. Most summarization techniques are based on extractive methods. Abstractive method is similar to summaries made by humans. Abstractive summarization as of now requires heavy machinery for language generation and is difficult to replicate into the domain specific areas.

FUTURE ENHANCEMENT

Currently in the project we are using unsupervised algorithm for clustering. We have used K-means clustering algorithm. Alternatively there are many other algorithms that can be used like latent Dirichlet Allocation algorithm. As a future enhancement, semi supervised algorithms can be used. Semi-supervised learning is a class of machine learning tasks and techniques that also make use of unlabelled data for training – typically a small amount of labelled data with a large amount of unlabelled data. There are many semi-supervised algorithms available like co-training. The dataset used in this project has labelled data that is data is classified according to topics. Another future enhancement could be we could take all data available and then have a algorithm to extract data relevant to topics that is use unlabelled data set.

Many other enhancements which can be done are :

A. Multiple document summarize can be achieved

B. It can be further expanded for multiple languages.

REFERENCES

- [1] **Title** : “Learning Python the Hard Way”
Author(s) : Zed Shaw
- [2] **Title** : “Performing literature review using text mining, Part III: Summarizing articles”
Author(s) : Dazhi Yang_ and Allan N. Zhang
Source(s) : IEEE Xplore Digital Library
- [3] **Title** : “Overview of extractive text summarization”
Author(s) : Ali Toofanzadeh Mozhdehi, Mohamad Abdolahi and Shohreh Rad Rahimi
Source(s) : IEEE Xplore Digital Library
- [4] **Title** : “Text Rank algorithm by exploiting Wikipedia for short text keywords extraction”
Author(s) : Wengen Li and Jiabao Zhao
Source(s) : IEEE Xplore Digital Library

Websites

- [5] <https://www.google.com/amp/s/nlpforhackers.io/textrank-text-summarization/amp>
- [6] <https://medium.com/the-artificial-impostor/use-textrank-to-extract-most-important-sentences-in-article-b8efc7e70b4>
- [7] <https://kdnuggets.com/2019/01/approaches-text-summarization-overview.html>
- [8] <https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/>