

A  
SEMINAR REPORT  
ON  
**Text Summarization using NLP**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING  
INFORMATION TECHNOLOGY

**BY**

Parth Vijay Raut  
Roll No: 33359  
Exam Seat No:

Under the guidance of  
Mr. Ravindra B. Murumkar



DEPARTMENT OF INFORMATION TECHNOLOGY  
PUNE INSTITUTE OF COMPUTER TECHNOLOGY  
SR. NO 27, PUNE-SATARA ROAD, DHANKAWADI  
PUNE - 411 043.  
AY: 2022-2023

SCTR's PUNE INSTITUTE OF COMPUTER TECHNOLOGY  
DEPARTMENT OF INFORMATION TECHNOLOGY



**C E R T I F I C A T E**

This is to certify that the Seminar work entitled  
**Text Summarization using NLP**

Submitted by

Name : Parth Vijay Raut

Exam Seat No: .

is a bonafide work carried out under the supervision of Mr. Ravindra B. Murumkar and it is submitted towards the partial fulfillment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Information Technology).

Mr. Ravindra Murumkar  
Seminar Guide

Dr. A. S. Ghotkar  
HOD IT

Dr. S. T. Gandhe  
Principal

Date:  
Place:

# **Acknowledgement**

I have learned a lot of new things and have delved deeply into the field of natural language processing thanks to this seminar project.

I would like to thank Dr. S. T. Gandhe, the principal, and A. S. Ghotkar, the head of the information technology department, of the Pune Institute of Computer Technology for their assistance in finishing my seminar report on text summarization using NLP.

My sincere appreciation goes out to our mentor, Mr. Ravindra Murumkar Sir, for his time and assistance with the seminar assignment. Your insightful counsel and recommendations were quite beneficial to me as I finished the assignment. I will always be grateful to you for this.

I also like to express my gratitude to Mr. Amit Kadam Sir for reviewing our seminar project and providing valuable insights.

Name :Parth Vijay Raut

Exam Seat No:

## Abstract

Text data is a precious source of information and knowledge and needs to be summarized in order to retain its most valid features. Summarization should not affect the semantic structure of the text. Text summarization can be categorized into two methods namely extractive and abstractive. Extractive summarization selects the sentences from the original text corpus and generates a summary while abstractive summarization interprets the original text corpus and generates the summary in its own words. While extractive summarization can be performed using TextRank which is an unsupervised-graph based technique, abstractive summarization can be performed using complex models such as the Seq2Seq model or the BERT model. Sequence-to-Sequence (Seq2Seq) modelling is about training the models that can convert sequences from one domain to sequences of another domain, Seq2Seq modelling is performed by the LSTM encoder and decoder. BERT, which stands for Bidirectional Encoder Representations from Transformers, is a Transformer, a deep learning model where every output element is connected to every input element.

**Keywords:** Abstractive summarization; Natural language processing; Text summarization; Extractive summarization; Deep learning; RNNs; Supervised learning; Transformers; BERT;

# Contents

|  |           |
|--|-----------|
| Certificate . . . . .  | i         |
| Acknowledgement . . . . .  | ii        |
| Abstract . . . . .   | iii       |
| Contents . . . . .   | iv        |
| List of Figures . . . . .  | v         |
| List of Tables . . . . .   | vi        |
| Abbreviations . . . . .  | vii       |
| <b>1 Introduction</b>  | <b>1</b>  |
| 1.1 Introduction . . . . .   | 1         |
| 1.2 Motivation . . . . .   | 1         |
| 1.3 Objectives . . . . .   | 1         |
| 1.4 Scope . . . . .  | 2         |
| <b>2 Literature Survey</b>   | <b>3</b>  |
| <b>3 Methodologies</b>   | <b>4</b>  |
| 3.1 Framework/Basic Architecture . . . . .                         | 4         |
| 3.2 Different Approaches . . . . .                                 | 5         |
| 3.3 State-of-the-art Algorithms . . . . .                          | 6         |
| <b>4 Implementation</b>  | <b>7</b>  |
| 4.1 Algorithm/Methodologies . . . . .                              | 7         |
| 4.2 Proposed Solution . . . . .                                    | 7         |
| 4.3 Software Requirement Specification . . . . .                   | 7         |
| 4.3.1 Platform for Implementation and its Specifications . . . . . | 7         |
| 4.4 Result . . . . .   | 8         |
| <b>5 Applications</b>  | <b>11</b> |
| 5.1 Applications . . . . .   | 11        |
| <b>6 Conclusion and Future Scope</b>                               | <b>12</b> |
| <b>References</b>  | <b>13</b> |

# List of Figures

|     |                                 |    |
|-----|---------------------------------|----|
| 3.1 | Methodology. . . . .            | 4  |
| 3.2 | RNNs. . . . .                   | 5  |
| 3.3 | LSTMs. . . . .                  | 5  |
| 3.4 | The Transformer model . . . . . | 6  |
| 4.1 | Implementaion 1 . . . . .       | 8  |
| 4.2 | Implementaion 2 . . . . .       | 8  |
| 4.3 | Implementaion 3 . . . . .       | 9  |
| 4.4 | Implementaion 4 . . . . .       | 9  |
| 4.5 | Implementaion 5 . . . . .       | 9  |
| 4.6 | Implementation 6 . . . . .      | 10 |
| 4.7 | Implementaion 7 . . . . .       | 10 |
| 4.8 | Implementaion 8 . . . . .       | 10 |

# List of Tables

|                                 |   |
|---------------------------------|---|
| 2.1 Literature Survey . . . . . | 3 |
|---------------------------------|---|

## **Abbreviations**

NLP : Natural Language Processing

LSTM : Long-short Term Memory

RNNs : Recurrent Neural Networks

BERT : Bidirectional Encoder Representations from Transformers



# 1. Introduction

## 1.1 Introduction

The amount of textual material on the web and other libraries is growing tremendously daily. Information utilization has become an expensive and time-consuming activity since data expands in a large quantity at a time and includes irrelevant content or noise. Text summarization is a method used to summarize the data. A manual text summarization process is undoubtedly an effective way to preserve the meaning of the text; however, this is a time-consuming activity. Another approach is to utilize the automatic text summarization (ATS). In ATS, different practical algorithms can be programmed into computers to produce summaries of information. Automatic text summarization is a method of evaluating, comprehending, and extracting information from human language.

## 1.2 Motivation

2.5 quintillion bytes of data is generated every single day and storage units are limited. It is impossible to store such huge amounts of data. Text summarization can extensively reduce this load on storage units by generating meaningful short summarized pieces of information. It is an important area of exploration as humans also tend to receive small pieces of information better thus making it a significant field of study.

## 1.3 Objectives

The main objective of a text summarization system is to identify the most useful from the given text and present it to the end users. Text summarization takes a large corpus of data as an input and outputs a meaningful and precise summaries of these corpuses. Other objectives of text summarization are listed as follows-

1. To identify large documents and conveniently convert them to meaningful summaries.
2. To study existing technologies and methods to perform the same.

## **1.4 Scope**

The scope of this seminar report is to study the different summarization techniques that include abstractive and extractive methods and the ways to implement these techniques in real life environments using python.

NLTK library from python is used for implementation and study of various extractive methods and abstractive methods is done.

## 2. Literature Survey

| Serial No. | Paper   | Method Used  |
|------------|---|--|
| 1          | Abdel-Salam, S.; Rafea, A. Performance Study on Extractive Text Summarization using BERT Models. Information 2022, 13, 67                           | In this research paper the author used different architectures that have been trained and evaluated for extractive summarization. They have used BERTSum as their baseline model and have also modified the source code, which is available in the BERTSum paper.  |
| 2          | M. F. Mridha, A. A. Lima, K. Nur, S. C. Das, M. Hasan and M. M. Kabir, "A Survey of Automatic Text Summarization: Progress, Process and Challenges, | This is a study that provides an overview of current research on NLP and ATS to quickly gain knowledge about NLP and ATS. Furthermore, it enables the creation of new tools, methods, datasets and resources that meet the needs of research and industry sectors. |
| 3          | P. Mahalakshmi and N. S. Fatima, "Summarization of Text and Image Captioning in Information Retrieval Using Deep Learning Techniques,"              | The research paper has proposed a summarization of Text and Image Captioning in Information Retrieval Using Deep Learning Techniques. The method proposed have used BiLSTM with the DBN model for the text summarization and image captioning process.             |

**Table 2.1:** Literature Survey

### 3. Methodologies

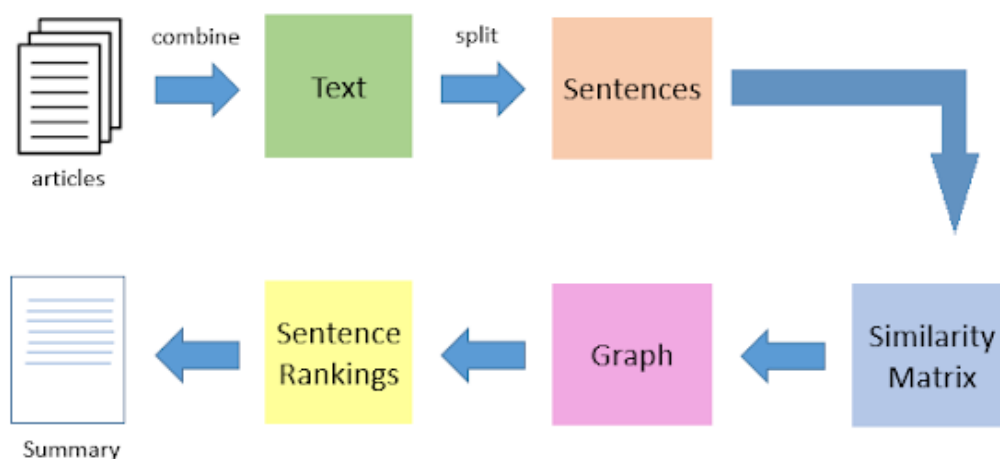
#### 3.1 Framework/Basic Architecture

**Extractive summarization** methods work just like that. It takes the text, ranks all the sentences according to the understanding and relevance of the text, and presents you with the most important sentences.

**Abstractive summarization**, on the other hand, tries to guess the meaning of the whole text and presents the meaning to you. It creates words and phrases, puts them together in a meaningful way, and along with that, adds the most important facts found in the text.

The following steps describe the working for text summarization

- The first step would be to concatenate all the text contained in the articles.
- Then split the text into individual sentences.
- In the next step, we will find vector representation (word embeddings) for each and every sentence.
- 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.
- Finally, a certain number of top-ranked sentences form the final summary



**Figure 3.1:** Methodology.

## 3.2 Different Approaches

### RNNs(Recurrent Neural Networks)

In RNN, new versions depend on previous versions. Because of this property of RNNs, we try to summarize the text as humanly as possible. Training: An iterative neural network uses a backpropagation algorithm, but applied to each timestamp. This is commonly known as backpropagation through time (BTT). A Sequence2Sequence model consisting of encoders and decoders is used to consolidate long input statements into one condensed line. These encoders and decoders also include LSTMs (see below) as an extension of simple RNNs. Has been updated.

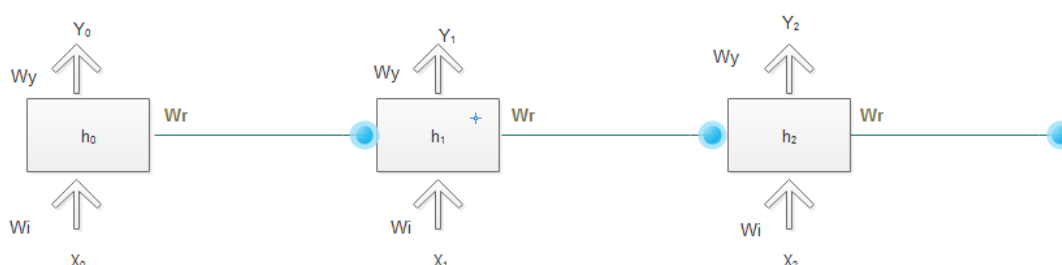


Figure 3.2: RNNs.

### LSTMs(Long-Short term memory network)

This is a special kind of RNN. It can learn long-term dependencies, so it can handle short-term dependency problems for simple RNNs. While trying to achieve this with an LSTM, the RNN cannot remember the context behind the input.

The tanh layer is used as a squashing function to convert the value to  $[-1,1]$ .

Identifies information here that is unnecessary and discarded from the cell's status. This decision is made by a sigmoid layer (as shown in the figure), also called a forgetting layer. The next sigmoid layer is trying to decide what new information to store.

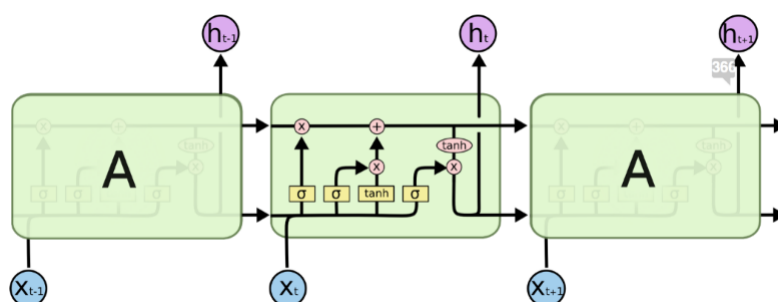


Figure 3.3: LSTMs.

### 3.3 State-of-the-art Algorithms

Google created the Transformer architecture in late 2018 to overcome issues with recurrent neural networks (RNNs). The RNN had two or more mistakes. Every memory cell in the

The following memory cell in line received RNN (see Figure 1). As a result, the model often "forgets" what came before in lengthy sequences. Only brief text snippets could be loaded into memory for training because RNNs are not very memory efficient.

In actuality, a stack of encoders is followed by a stack of decoders in the Transformer architecture. The encoder/decoder stack is depicted in Figure. A self-awareness layer and a feedforward layer are both present in every coding layer. The transformer's function is to convert a string of words into a vector that is equally weighted. The decoder tries to reconstruct the words that have been obscured in the input by masking some words (or a new set of words, as in a translation task). The decoder operates similarly, with the exception that between the attention layer and the feedforward layer there is additionally an encoder/decoder attention layer. In order to get the masked) to anticipate the word, the decoder can learn which words in the sequence to look at. (See Figure)

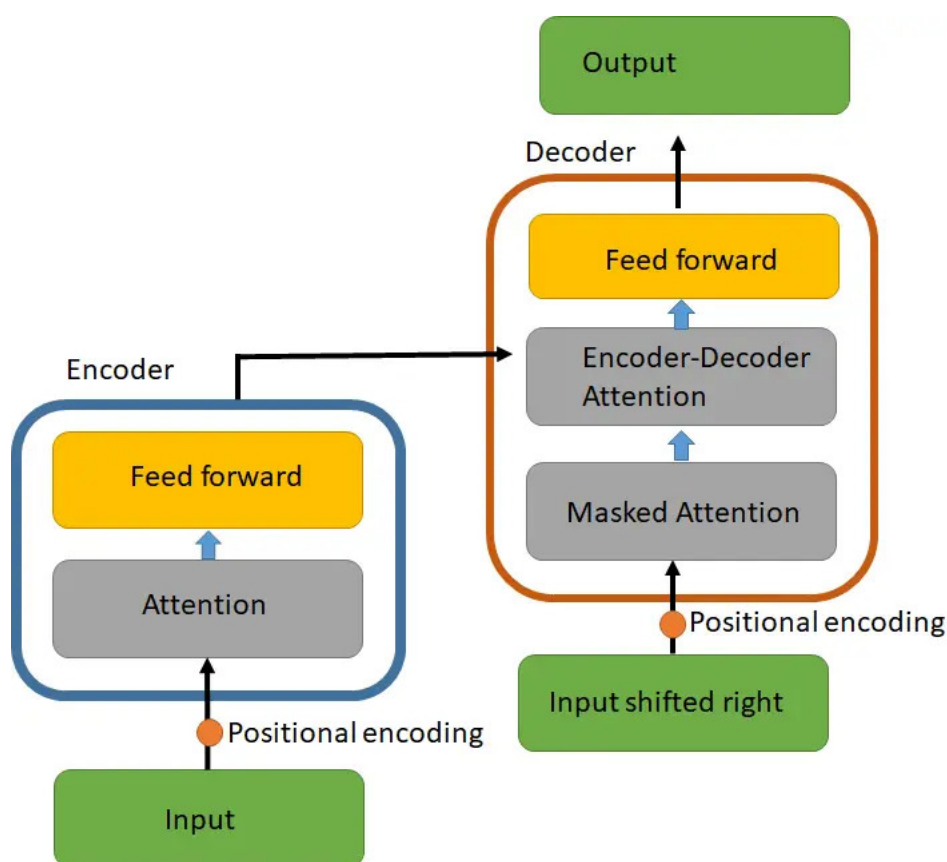


Figure 3.4: The Transformer model

## 4. Implementation

### 4.1 Algorithm/Methodologies

Natural Language Processing (NLP) is a process of manipulating or understanding the text or speech by any software or machine. An analogy is that humans interact and understand each other's views and respond with the appropriate answer. In NLP, this interaction, understanding, and response are made by a computer instead of a human.

### 4.2 Proposed Solution

Task 1 - Convert the input text to a list of sentences. Then, compute the number of sentences in the given Text.

Task 2 - Calculate the frequency of words in each sentence: the output should be a dictionary where each key is a sentence and the value is also a dictionary of word frequency.

Task 3 - Calculate Term frequency for each word in a sentence.  $TF(\text{word}) = (\text{Number of times term "word" appears in a sentence}) / (\text{Total number of terms in the sentence})$

Task 4 - Create a matrix termFrequency: The termFrequency matrix should be a dictionary where each key is a sentence and the value is also a dictionary of word frequency.

Task 5 - For each word compute how many sentences contain that word.

Task 6- Calculate IDF for each word in a sentence.  $IDF(\text{word}) = \log e(\text{Total number of sentences} / \text{number of sentences with term word in it})$

Task 7 - Compute the TF-IDF for each word in each sentence.

Task 8 - Generate the summary : select a sentence for summarization if the weight of the sentence exceeds the threshold

### 4.3 Software Requirement Specification

Google colab or Jupyter notebook

#### 4.3.1 Platform for Implementation and its Specifications

Memory and disk space required per user: 1GB RAM + 1GB of disk + . 5 CPU core.

Server overhead: 2-4GB or 10percent system overhead (whatever is larger), . 5 CPU cores,

10GB disk space.

Port requirements: Port 8000 plus 5 unique, random ports per notebook.

Python 3

Libraries:

Natural Language Tool kit

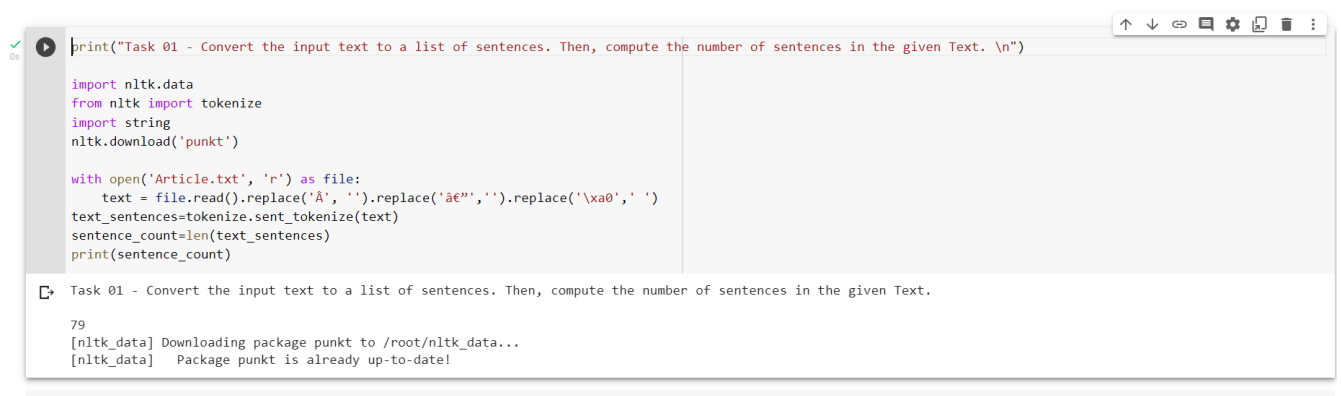
SpaCy

String

NumPy

## 4.4 Result

### ▼ Text Summarization using Python-NLTK



```
print("Task 01 - Convert the input text to a list of sentences. Then, compute the number of sentences in the given Text. \n")

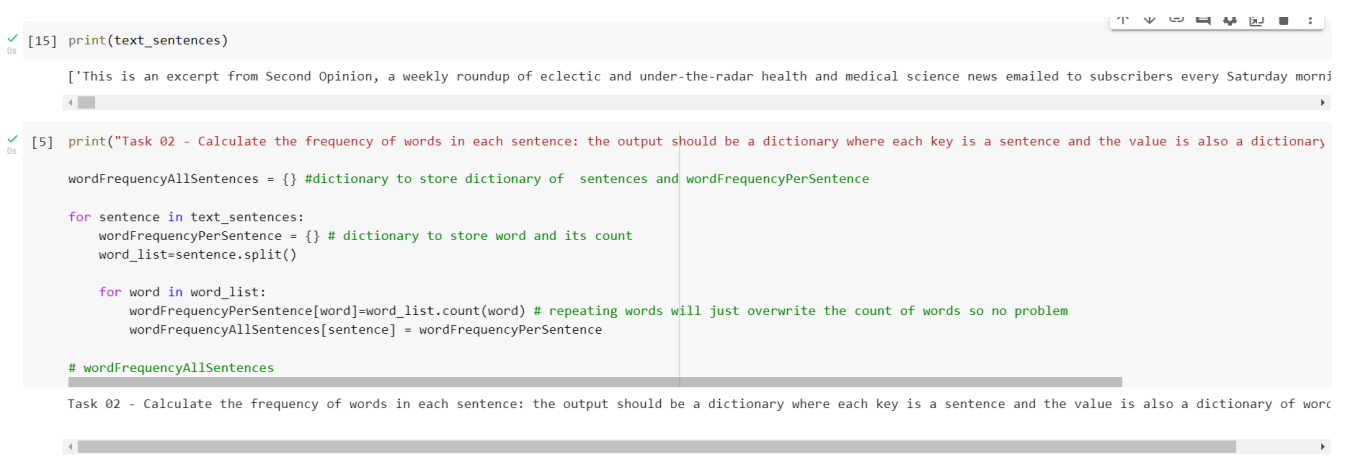
import nltk.data
from nltk import tokenize
import string
nltk.download('punkt')

with open('Article.txt', 'r') as file:
    text = file.read().replace('A', '').replace('ä€', '').replace('\xa0', ' ')
text_sentences=tokenize.sent_tokenize(text)
sentence_count=len(text_sentences)
print(sentence_count)

Task 01 - Convert the input text to a list of sentences. Then, compute the number of sentences in the given Text.

79
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
```

Figure 4.1: Implementaion 1



```
[15] print(text_sentences)

['This is an excerpt from Second Opinion, a weekly roundup of eclectic and under-the-radar health and medical science news emailed to subscribers every Saturday morni']

[5] print("Task 02 - Calculate the frequency of words in each sentence: the output should be a dictionary where each key is a sentence and the value is also a dictionary")

wordFrequencyAllSentences = {} #dictionary to store dictionary of sentences and wordFrequencyPerSentence

for sentence in text_sentences:
    wordFrequencyPerSentence = {} # dictionary to store word and its count
    word_list=sentence.split()

    for word in word_list:
        wordFrequencyPerSentence[word]=word_list.count(word) # repeating words will just overwrite the count of words so no problem
        wordFrequencyAllSentences[sentence] = wordFrequencyPerSentence

# wordFrequencyAllSentences

Task 02 - Calculate the frequency of words in each sentence: the output should be a dictionary where each key is a sentence and the value is also a dictionary of words
```

Figure 4.2: Implementaion 2



```

✓ ▶ print("Task 03 - Calculate Term frequency for each word in a sentence. \n")

from nltk import word_tokenize

TFwordAllSentences = {} #dictionary to store dictionary of sentences and TFWordPerSentence

for sentence in text_sentences:
    TFWordPerSentence = {} # dictionary to store word and TF(word)
    # word_list=sentence.split()
    word_list=word_tokenize(sentence.lower())
    word_list=[word.lower() for word in word_list if word.isalpha()] # removing punctuations

    for word in word_list:
        TFWordPerSentence[word]=word_list.count(word) / len(word_list)
    TFwordAllSentences[sentence] = TFWordPerSentence

# TFwordAllSentences

```

Task 03 - Calculate Term frequency for each word in a sentence.

Figure 4.3: Implementaion 3

```

✓ [7] print("Task 04 - Create a matrix termFrequency: The termFrequency matrix should be a dictionary where each key is a sentence and the value is also a dictionary of word frequency. \n")

# already calculated in 1.2
termFrequencyMatrix = wordFrequencyAllSentences;
# termFrequencyMatrix

```

Task 04 - Create a matrix termFrequency: The termFrequency matrix should be a dictionary where each key is a sentence and the value is also a dictionary of word frequency.

```

✓ ▶ print("Task 05 - For each word compute how many sentences contain that word. \n")

from nltk import word_tokenize
allWords = word_tokenize(text.lower())

allWords=[word.lower() for word in allWords if word.isalpha()] # removing punctuations
allWords=set(allWords) # getting unique words in text

word_count= dict()

for word in allWords:
    count=0
    for sentence in text_sentences:
        if(word in sentence.lower()): # check if word is present in sentence
            count=count+1 # count sentene
    word_count[word]=count

# word_count

```

Task 05 - For each word compute how many sentences contain that word.

Figure 4.4: Implementaion 4

```

✓ ▶ print("Task 06 - Calculate IDF for each word in a sentence. \n")

import math

IDFwordsAllSentences = {} #dictionary to store dictionary of sentences and IDFwordPerSentence
totalSentences=len(text_sentences) ## for using in formula

for sentence in text_sentences:
    IDFwordPerSentence = {} # dictionary to store word and its IDF

    word_list=word_tokenize(sentence.lower())
    word_list=[word.lower() for word in word_list if word.isalpha()] # removing punctuations

    for word in word_list:
        if(word_count[word]==0):
            pass
        else:
            IDFwordPerSentence[word]= math.log10(totalSentences / word_count[word]) # repeating words will just overwrite the count of words so no problem
    IDFwordsAllSentences[sentence] = IDFwordPerSentence

# IDFwordsAllSentences

```

Task 06 - Calculate IDF for each word in a sentence.

Figure 4.5: Implementaion 5

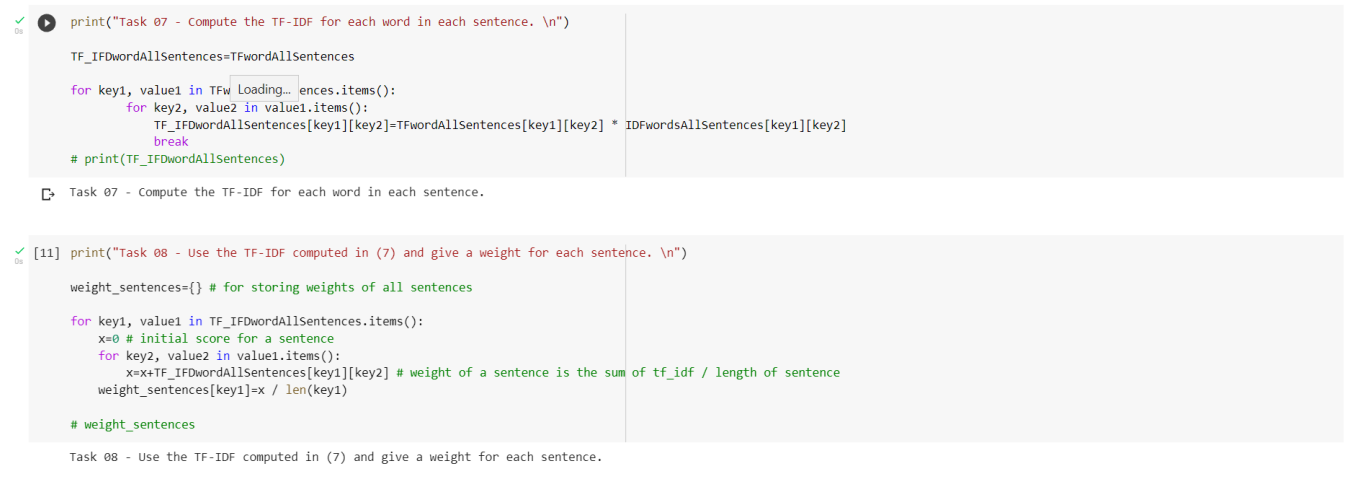


Figure 4.6: Implementation 6

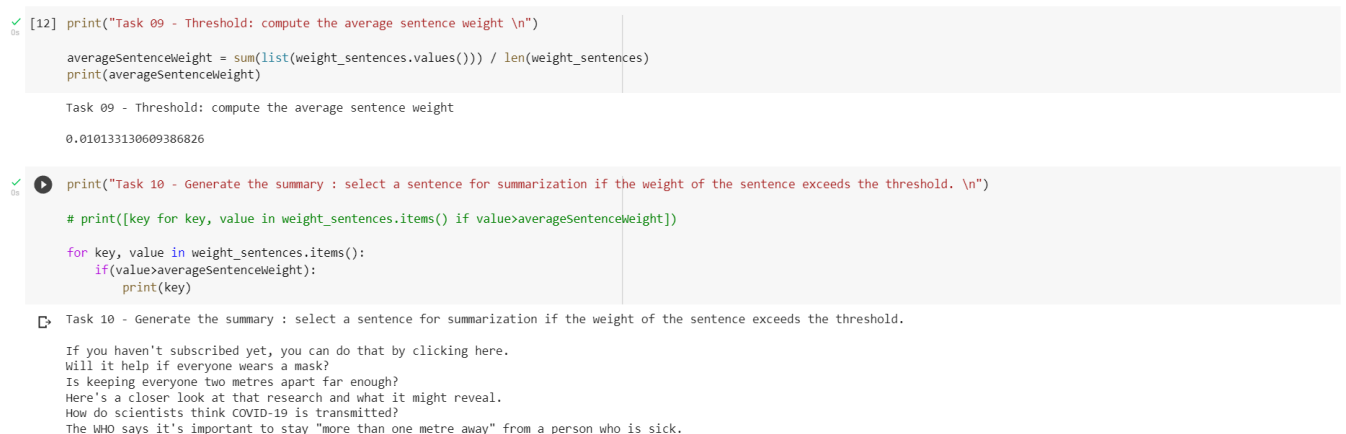


Figure 4.7: Implementaion 7

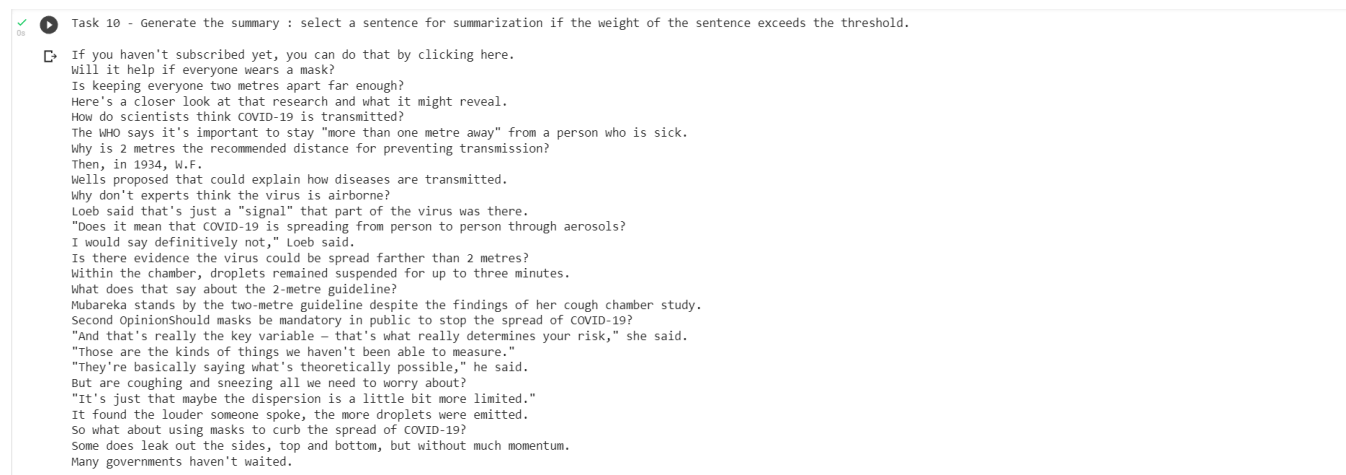


Figure 4.8: Implementaion 8

## **5. Applications**

### **5.1 Applications**

Summarization is the task of condensing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content.

Text summarization can be used by personal or specialised assistants, that would be the good use case. Apart from that it can be used for many personalized devices or applications. Mail clients, report generation, news feed etc.

Some important applications of Text summarization are listed below

- Book or Novel Summarization
- Tweet Summarization
- Entity centric Summarization
- News Summarization
- Email Summarization
- Document Summarization
- Summarization of scientific papers

## **6. Conclusion and Future Scope**

Although text summarising is an old issue, academics are still becoming interested in it. However, overall text summarising performance is only average, and the summaries produced are not always accurate. Researchers are therefore working to enhance current text-summarizing techniques. It's also a top priority to create summaries that are resilient and of higher quality using unique summarising techniques. In order to perform better, ATS should be made more intelligent by fusing it with other integrated systems. In order to summarise and reduce the volume of text, automatic text summarization is a prominent area of research that is widely applied and integrated into various applications.

## Bibliography

- [1] Abdel-Salam, S.; Rafea, A. *Performance Study on Extractive Text Summarization using BERT Models*. *Information* 2022, 13, 67. <https://doi.org/10.3390/info13020067>
- [2] M. F. Mridha, A. A. Lima, K. Nur, S. C. Das, M. Hasan and M. M. Kabir, "A Survey of Automatic Text Summarization: Progress, Process and Challenges," in *IEEE Access*, vol. 9, pp. 156043-156070, 2021, doi: 10.1109/ACCESS.2021.3129786.
- [3] P. Mahalakshmi and N. S. Fatima, "Summarization of Text and Image Captioning in Information Retrieval Using Deep Learning Techniques," in *IEEE Access*, vol. 10, pp. 18289-18297, 2022, doi: 10.1109/ACCESS.2022.3150414.
- [4] M. Jang and P. Kang, "Learning-Free Unsupervised Extractive Summarization Model," in *IEEE Access*, vol. 9, pp. 14358-14368, 2021, doi: 10.1109/ACCESS.2021.3051237.