INTRODUCTION

Various methods to automatically summarize text corpus are available and also actively being developed as text summarization is an important field. This is especially important seeing that more and more text data is being developed every day at a high rate and the humans have only a limited time. Thus, reducing the reading time is a very desirable prospect as it can greatly decrease human effort as well as help in finding the most important parts of any document or corpus of documents. While summarizing the document, keeping the relevancy and redundancy under control is very important for any automatic summarizer.

Text summarization refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document.

1.1 Types of Text Summarization

There are two main types summarization based on its output in the area of NLP (natural language processing):

1.1.1 Extraction-based summarization:

The extractive text summarization technique involves identifying key phrases from the source document and combining them to make a summary. The extraction is made according to the defined metric without making any changes to the texts.

Here is an example:

Source text: *Joseph and Mary rode on a donkey to attend the annual event in Jerusalem. In the city, Mary gave birth to a child named Jesus.*

Extractive summary: *Joseph and Mary attend event Jerusalem. Mary birth Jesus.*

As seen above, the words already existing in the source document have been extracted and joined to create a summary — although sometimes the summary can be grammatically strange.

1.1.2 Abstraction-based summarization:

The abstraction technique entails paraphrasing and shortening parts of the source document. When abstraction is applied for text summarization in deep learning problems, it can overcome the grammar inconsistencies of the extractive method.
The abstractive text summarization algorithms create new phrases and sentences that relay the most useful information from the original text — just like humans do.
Therefore, abstraction performs better than extraction. However, the text summarization algorithms required to do abstraction are more difficult to develop; that's why the use of extraction is still popular.

Here is an example:
Abstractive summary: *Joseph and Mary came to Jerusalem where Jesus was born.*

1.2 Commonly Used Methods for Text Summarization

There are various techniques that have been applied to text summarization and each has its own advantage. One of the most commonly used and successful algorithms are those that are inspired by the nature. A few such examples are Genetic Algorithm (GA), Particle Swarm Algorithm (PSO), Ant Colony Optimization (ACO), Firefly Algorithm (FA), etc. These algorithms are explained below.

1.2.1 Genetic Algorithm

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

This notion can be applied for a text summarization problem. We consider a set of solutions for a problem and select the set of best ones out of them.

1.2.2 Particle Swarm Optimization

Particle Swarm Optimization (originally proposed to simulate birds searching for food, the movement of fishes' shoal, etc.) is able to simulate behaviors of swarms in order to optimize a numeric problem iteratively. It can be classified as a swarm intelligence algorithm like Ant Colony Algorithm, Artificial Bee Colony Algorithm and Bacterial Foraging, for example.

Consisting in the constant search of best solution, the method moves the particles (in this case represented as a (x,y) position) with a certain velocity calculated in every iteration. Each particle's movement has the influence of its own the best-known position and also the best-known position in the space-search. The final result expected is that the particle swarm converges to the best solution. It's important to mention that PSO doesn't use Gradient Descent, so it can be used to nonlinear problems once it doesn't require that the problem have to be differentiable.

At first, in the 2 for loops, it initializes the particles' positions with a random uniform distribution within a permissible range for all its dimensions (Some problems require handling to several dimensions). After that, for each particle, it calculates its fitness value and compared with his own best.

1.2.3 Ant Colony Optimization

Ant colony optimization (ACO) takes inspiration from the foraging behavior of some ant species. These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. Ant colony optimization exploits a similar mechanism for solving optimization problems.

The ants construct the solutions as follows. Each ant starts from a randomly selected city (vertex of the construction graph). Then, at each construction step it moves along the edges of the graph. Each ant keeps a memory of its path, and in subsequent steps it chooses among the edges that do not lead to vertices that it has already visited. An ant has constructed a solution once it has visited all the vertices of the graph. At each construction step, an ant probabilistically chooses the edge to follow among those that lead to yet unvisited vertices. The probabilistic rule is biased by pheromone values and heuristic information: the higher the pheromone and the heuristic value associated to an edge, the higher the probability an ant will choose that particular edge. Once all the ants have completed their tour, the pheromone on the edges is updated. Each of the pheromone values is initially decreased by a certain percentage. Each edge then receives an amount of additional pheromone proportional to the quality of the solutions to which it belongs (there is one solution per ant).

This procedure is repeatedly applied until a termination criterion is satisfied.

1.2.4 Firefly Algorithm

Firefly algorithm is one of the swarm-based algorithms which is easy to understand and use. Itwas introduced by Yang (2009) and is based on the behavior of fireflies. Nature has inspired many algorithms in the computer science world. Mostly, many meta-heuristic algorithms has been developed using many models found naturally. Natural selection and survival of the fittest is the main idea behind these algorithms. Many animals in the wild use various methods to communicate. Fireflies use the light or flashing property to do the same. The light is emitted by a process called bioluminescence. Usually, a short flash of light is produced by the fireflies. Naturally, the fireflies use this light to attract potential mates by emitting the same pattern of light. Since light intensity decreases with increase in distance, and also the visibility also decreases with the distance, this communication only works in certain a range and also decreases with increase in distance.

In the algorithm, the intensity of the flashing light is made so as it is directly proportional to the value of the fitness or objective function taken, which is generally independent of the algorithm and depends on the type of optimization to be done. Combining the swarm intelligence with a suitable fitness function leads to good optimization performances.

1.2.5 Run from Localhost

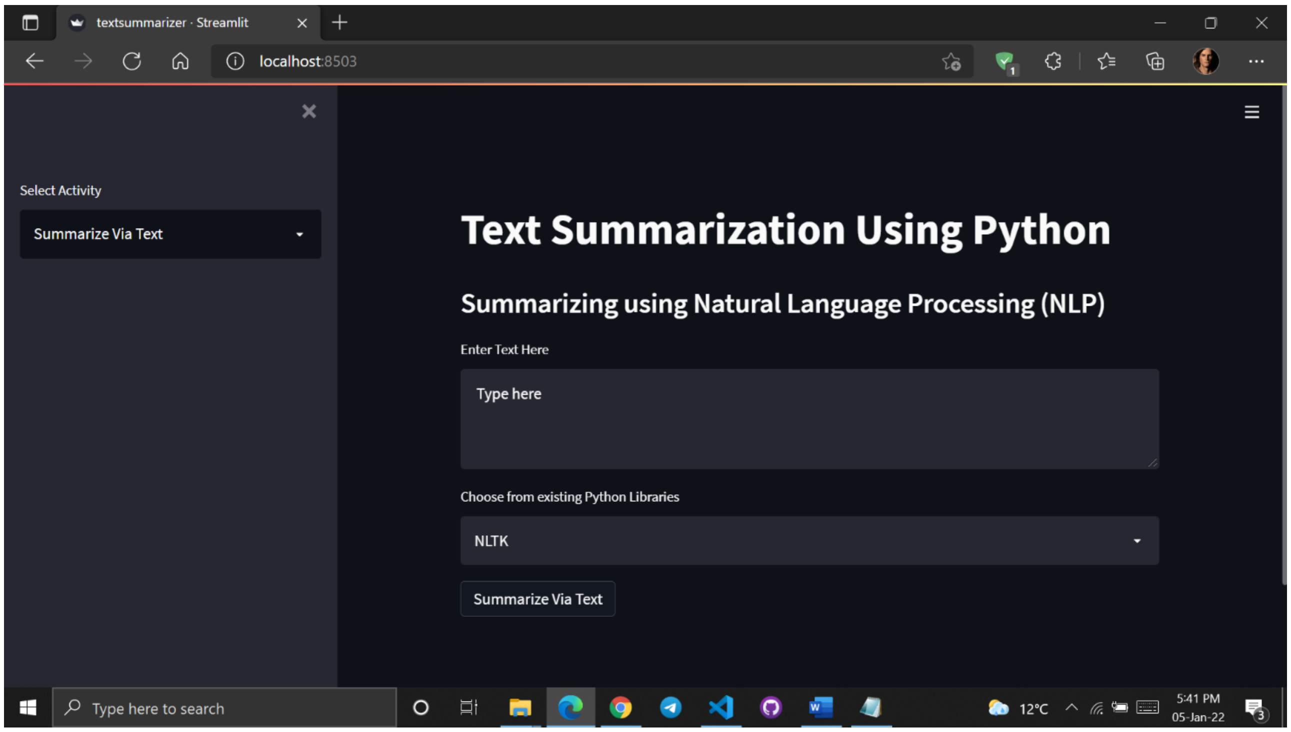
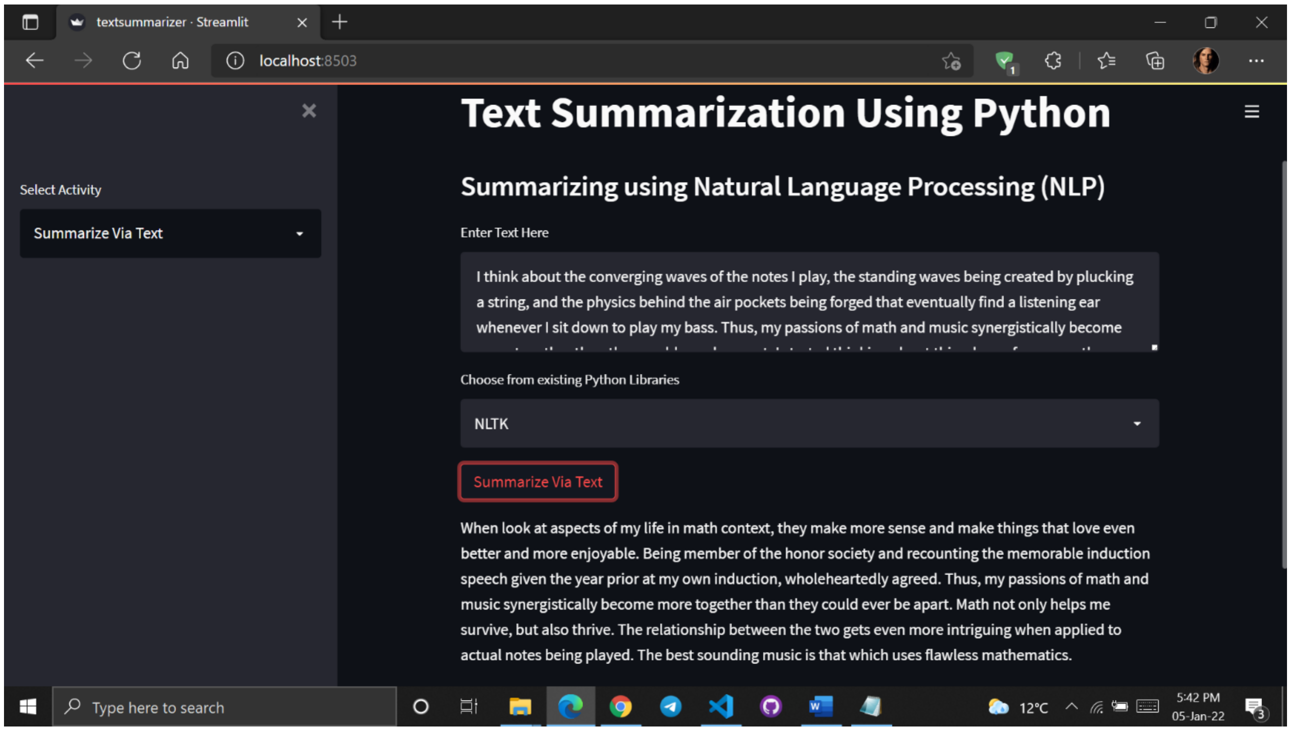

textsummarizer · Streamlit
localhost:8503
Select Activity
Summarize Via Text
Text Summarization Using Python
Summarizing using Natural Language Processing (NLP)
Enter Text Here
Type here
Choose from existing Python Libraries
NLTK
Summarize Via Text
Type here to search
12°C
5:41 PM
05-Jan-22

Fig. 1 Text Summarization Using Python's Natural Language Processing



Text Summarization Using Python

Summarizing using Natural Language Processing (NLP)

Enter Text Here

I think about the converging waves of the notes I play, the standing waves being created by plucking a string, and the physics behind the air pockets being forged that eventually find a listening ear whenever I sit down to play my bass. Thus, my passions of math and music synergistically become

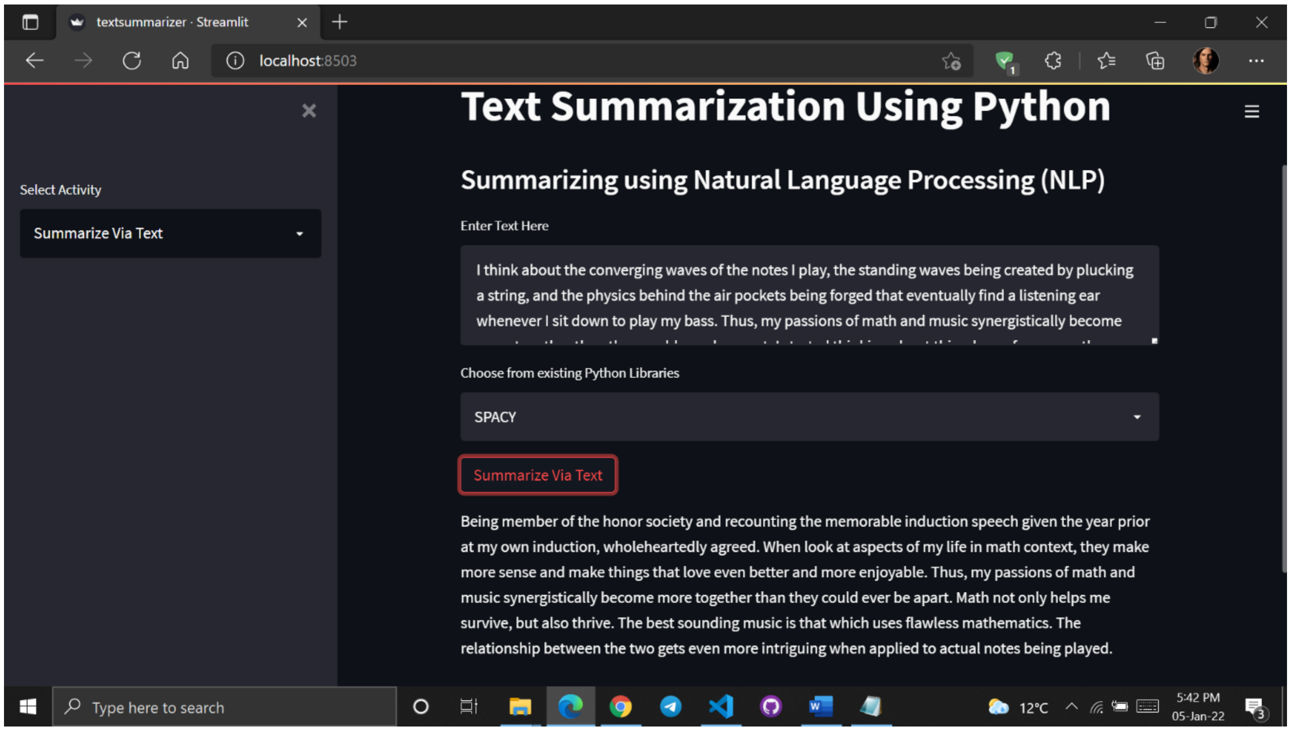
Choose from existing Python Libraries

NLTK

Summarize Via Text

When look at aspects of my life in math context, they make more sense and make things that love even better and more enjoyable. Being member of the honor society and recounting the memorable induction speech given the year prior at my own induction, wholeheartedly agreed. Thus, my passions of math and music synergistically become more together than they could ever be apart. Math not only helps me survive, but also thrive. The relationship between the two gets even more intriguing when applied to actual notes being played. The best sounding music is that which uses flawless mathematics.

Fig. 2 Text Summarization Using Python NLTK



Text Summarization Using Python

Summarizing using Natural Language Processing (NLP)

Enter Text Here

I think about the converging waves of the notes I play, the standing waves being created by plucking a string, and the physics behind the air pockets being forged that eventually find a listening ear whenever I sit down to play my bass. Thus, my passions of math and music synergistically become

Choose from existing Python Libraries

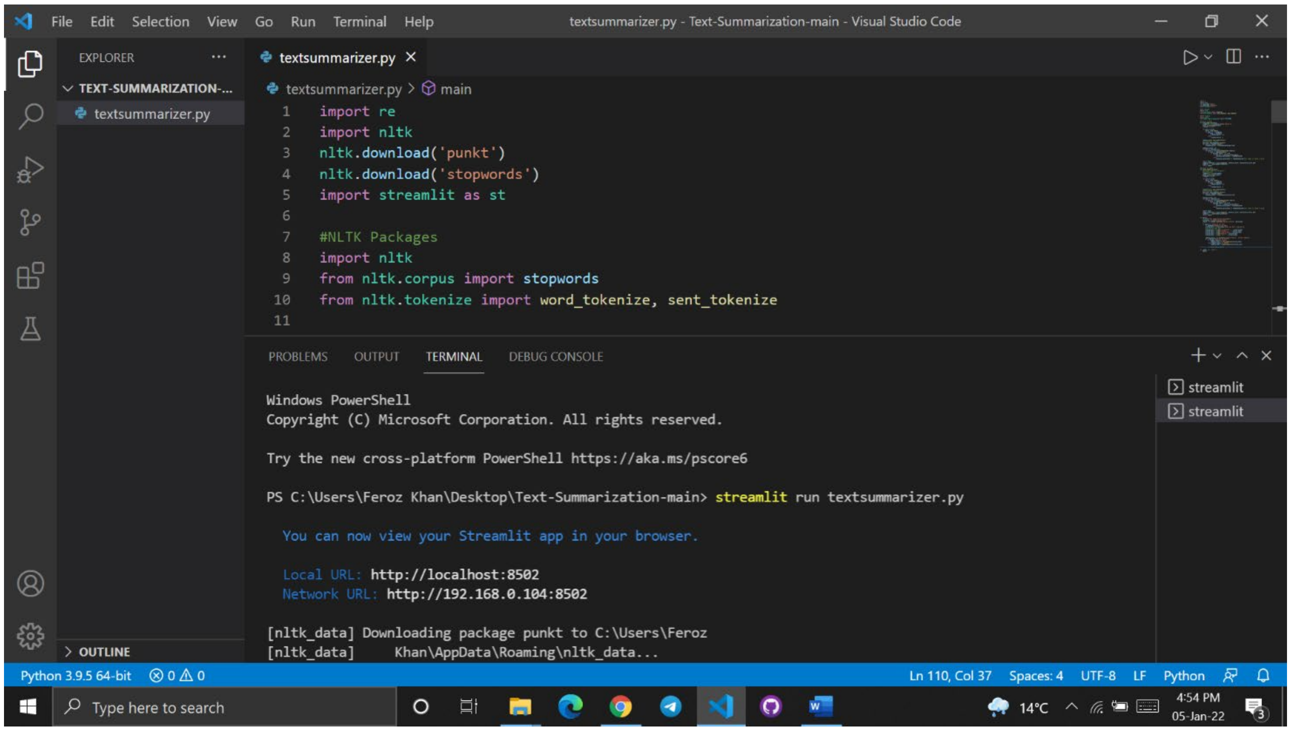
SPACY

Summarize Via Text

Being member of the honor society and recounting the memorable induction speech given the year prior at my own induction, wholeheartedly agreed. When look at aspects of my life in math context, they make more sense and make things that love even better and more enjoyable. Thus, my passions of math and music synergistically become more together than they could ever be apart. Math not only helps me survive, but also thrive. The best sounding music is that which uses flawless mathematics. The relationship between the two gets even more intriguing when applied to actual notes being played.

Fig. 3 Text Summarization Using Python SPACY

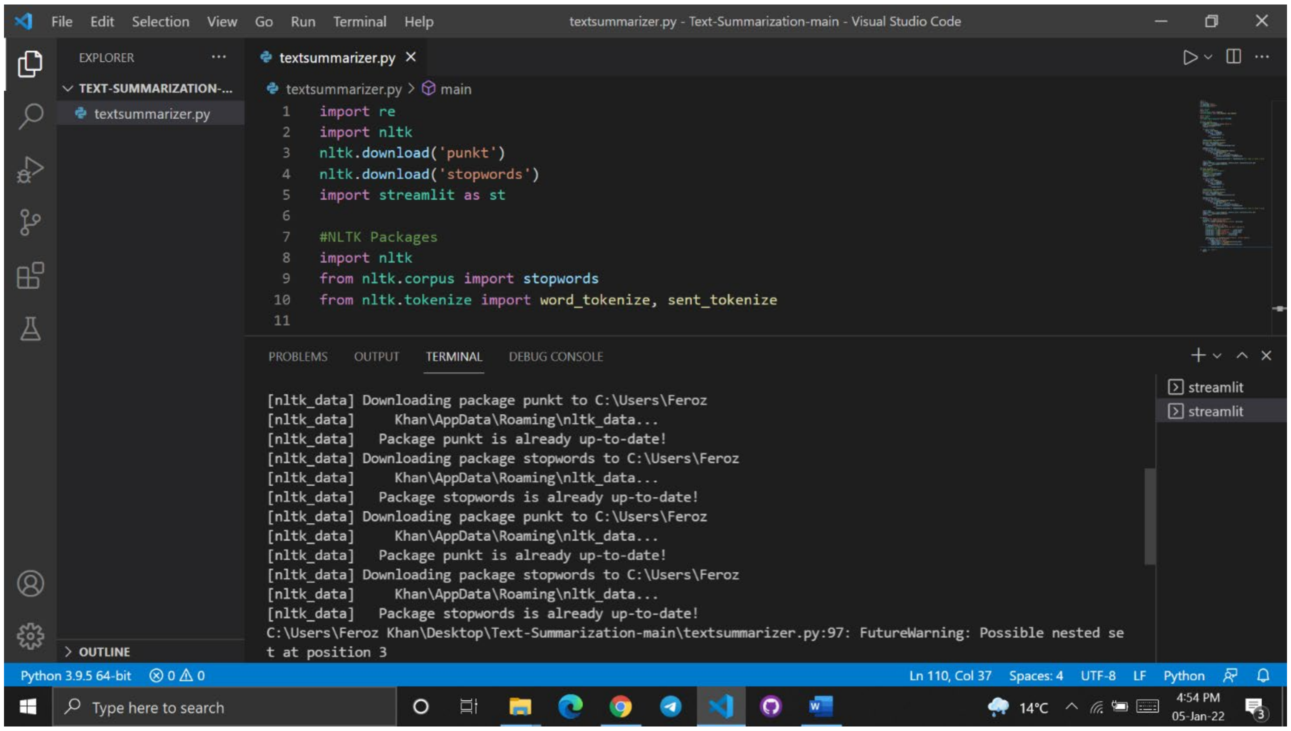
1.2.6 Command-line Interface


File Edit Selection View Go Run Terminal Help
textsummarizer.py - Text-Summarization-main - Visual Studio Code
EXPLORER
TEXT-SUMMARIZATION-...
textsummarizer.py
textsummarizer.py > main
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
streamlit
streamlit
OUTLINE
Python 3.9.5 64-bit ⊗ 0 ⚠ 0
Ln 110, Col 37 Spaces: 4 UTF-8 LF Python
Type here to search
14°C
4:54 PM
05-Jan-22

```python
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
import streamlit as st

#NLTK Packages
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Feroz Khan\Desktop\Text-Summarization-main> streamlit run textsummarizer.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.0.104:8502

[nltk_data] Downloading package punkt to C:\Users\Feroz
[nltk_data]     Khan\AppData\Roaming\nltk_data...
```

Fig. 4 Command for Execution of Program

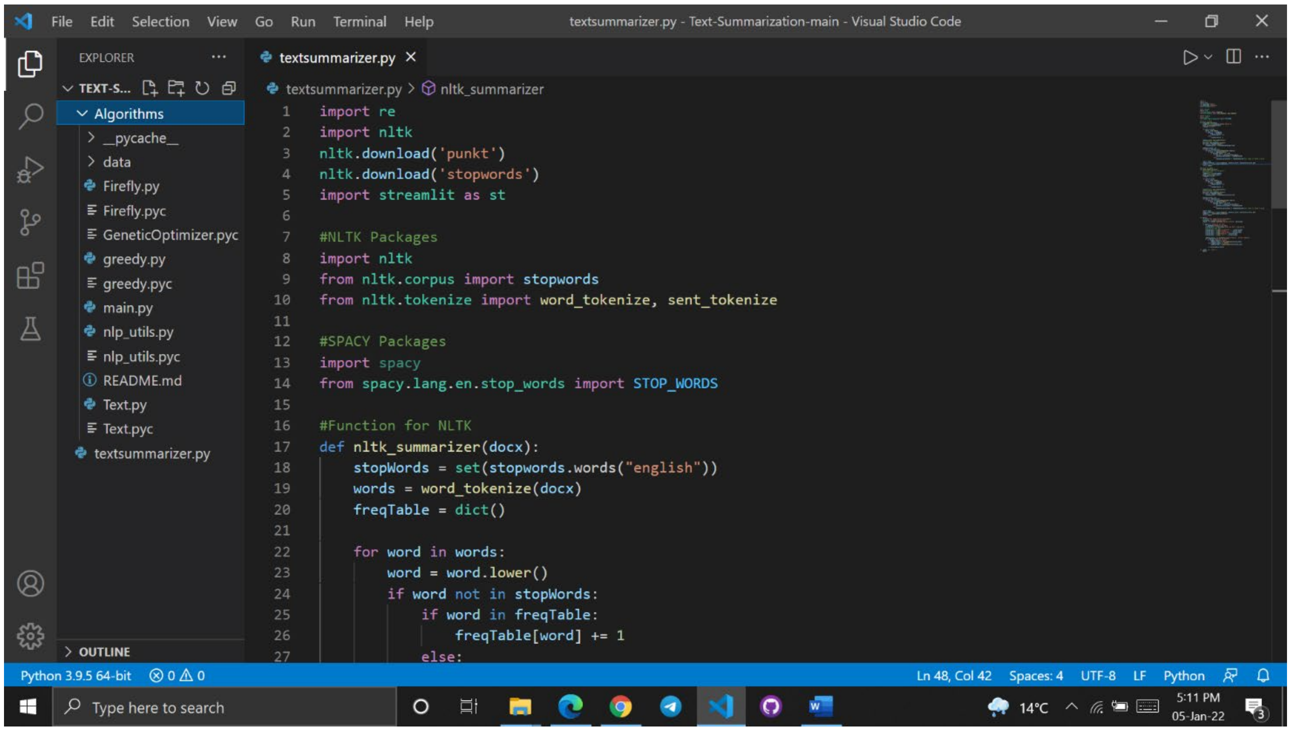

File Edit Selection View Go Run Terminal Help
textsummarizer.py - Text-Summarization-main - Visual Studio Code
EXPLORER
TEXT-SUMMARIZATION-...
textsummarizer.py
textsummarizer.py > main
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
streamlit
streamlit
OUTLINE
Python 3.9.5 64-bit ⊗ 0 ⚠ 0
Ln 110, Col 37 Spaces: 4 UTF-8 LF Python
Type here to search
14°C
4:54 PM
05-Jan-22

```python
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
import streamlit as st

#NLTK Packages
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

```

```
[nltk_data] Downloading package punkt to C:\Users\Feroz
[nltk_data]     Khan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\Feroz
[nltk_data]     Khan\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\Feroz
[nltk_data]     Khan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\Feroz
[nltk_data]     Khan\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
C:\Users\Feroz Khan\Desktop\Text-Summarization-main\textsummarizer.py:97: FutureWarning: Possible nested se
t at position 3
```

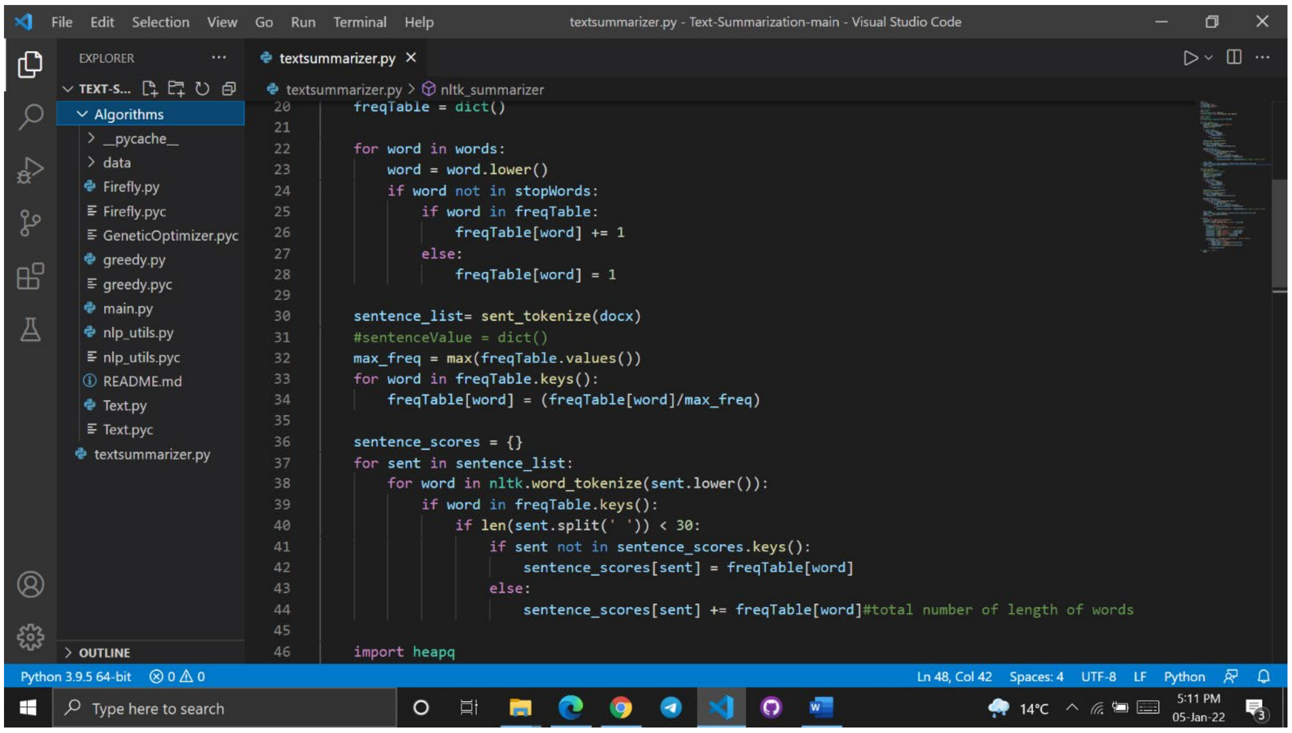
Fig. 5 Program Executing Successfully

1.2.6 Source Code of the Algorithms



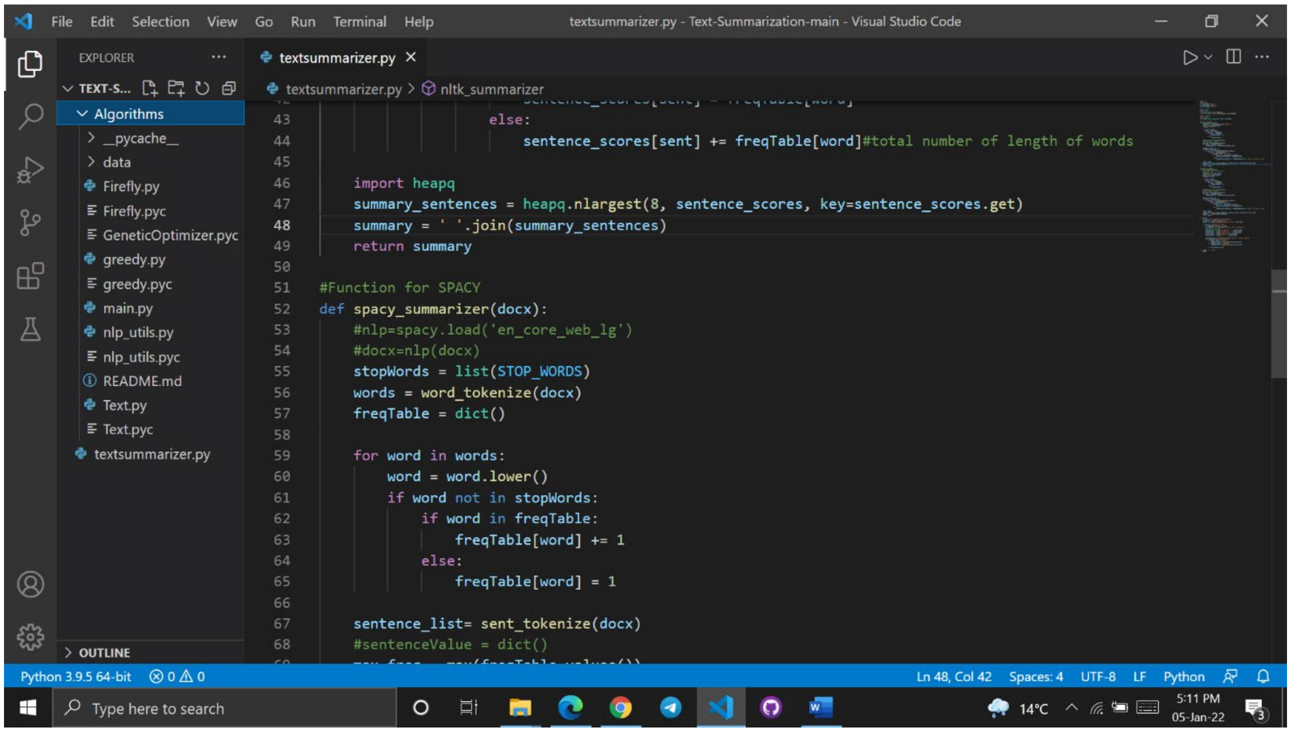
```python
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
import streamlit as st

#NLTK Packages
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

#SPACY Packages
import spacy
from spacy.lang.en.stop_words import STOP_WORDS

#Function for NLTK
def nltk_summarizer(docx):
    stopWords = set(stopwords.words("english"))
    words = word_tokenize(docx)
    freqTable = dict()

    for word in words:
        word = word.lower()
        if word not in stopWords:
            if word in freqTable:
                freqTable[word] += 1
            else:
```

Fig. 6



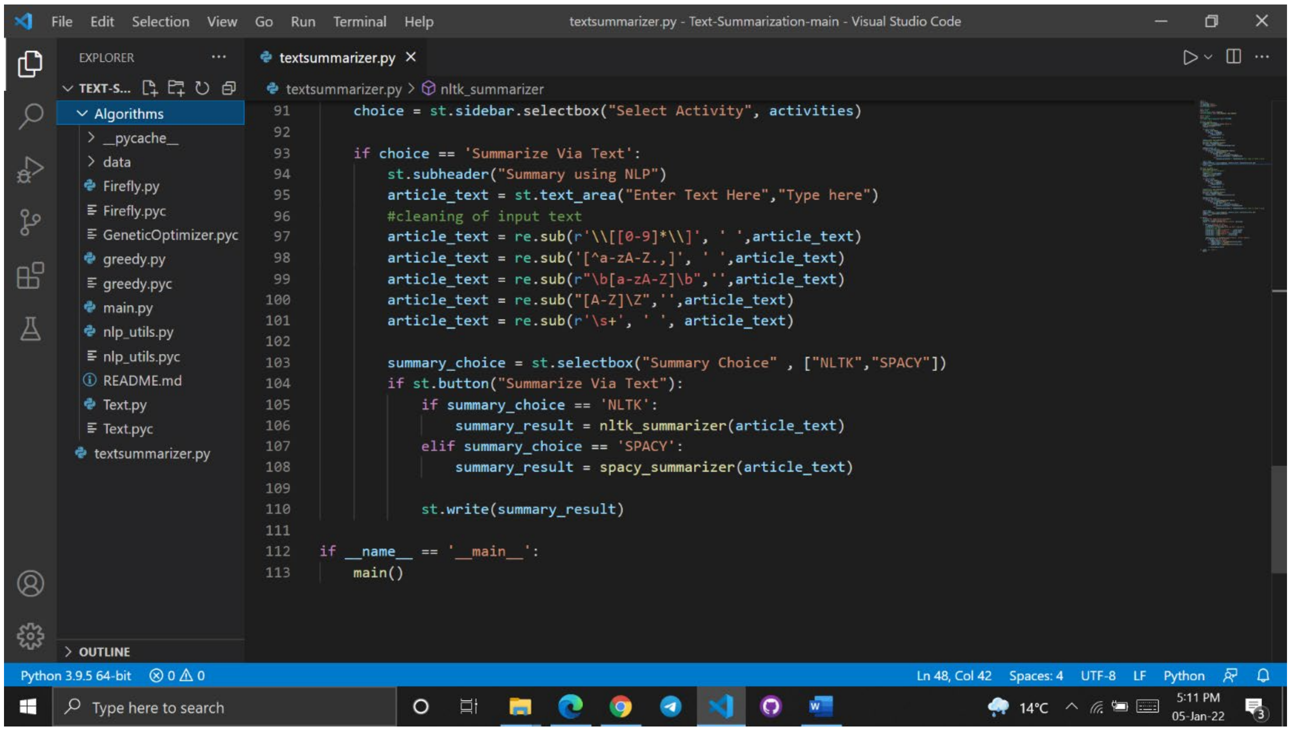
```python
    freqTable = dict()

    for word in words:
        word = word.lower()
        if word not in stopWords:
            if word in freqTable:
                freqTable[word] += 1
            else:
                freqTable[word] = 1

    sentence_list= sent_tokenize(docx)
    #sentenceValue = dict()
    max_freq = max(freqTable.values())
    for word in freqTable.keys():
        freqTable[word] = (freqTable[word]/max_freq)

    sentence_scores = {}
    for sent in sentence_list:
        for word in nltk.word_tokenize(sent.lower()):
            if word in freqTable.keys():
                if len(sent.split(' ')) < 30:
                    if sent not in sentence_scores.keys():
                        sentence_scores[sent] = freqTable[word]
                    else:
                        sentence_scores[sent] += freqTable[word]#total number of length of words

    import heapq
```

Fig. 7



```python
                    else:
                        sentence_scores[sent] += freqTable[word]#total number of length of words

    import heapq
    summary_sentences = heapq.nlargest(8, sentence_scores, key=sentence_scores.get)
    summary = ' '.join(summary_sentences)
    return summary

#Function for SPACY
def spacy_summarizer(docx):
    #nlp=spacy.load('en_core_web_lg')
    #docx=nlp(docx)
    stopWords = list(STOP_WORDS)
    words = word_tokenize(docx)
    freqTable = dict()

    for word in words:
        word = word.lower()
        if word not in stopWords:
            if word in freqTable:
                freqTable[word] += 1
            else:
                freqTable[word] = 1

    sentence_list= sent_tokenize(docx)
    #sentenceValue = dict()
```

Fig. 8



```python
    choice = st.sidebar.selectbox("Select Activity", activities)

    if choice == 'Summarize Via Text':
        st.subheader("Summary using NLP")
        article_text = st.text_area("Enter Text Here","Type here")
        #cleaning of input text
        article_text = re.sub(r'\\[[0-9]*\\]', ' ',article_text)
        article_text = re.sub('[^a-zA-Z.,]', ' ',article_text)
        article_text = re.sub(r"\b[a-zA-Z]\b",'',article_text)
        article_text = re.sub("[A-Z]\Z",'',article_text)
        article_text = re.sub(r'\s+', ' ', article_text)

        summary_choice = st.selectbox("Summary Choice" , ["NLTK","SPACY"])
        if st.button("Summarize Via Text"):
            if summary_choice == 'NLTK':
                summary_result = nltk_summarizer(article_text)
            elif summary_choice == 'SPACY':
                summary_result = spacy_summarizer(article_text)

            st.write(summary_result)

if __name__ == '__main__':
    main()
```

Fig. 9