## MINI PROJECT REPORT

**(2019-20)**

TEXT SUMMARIZER



# Institute of Engineering & Technology

Submitted by

Afreen Hasan (171500022)

***Supervised By***

Mr. Vinay Agrawal

Assistant Professor

**Department of Computer Engineering & Applications**

# ABSTRACT

Automatic text summarization is basically summarizing of the given paragraph using natural language processing and machine learning. There has been an explosion in the amount of text data from a variety of sources. This volume of text is an invaluable source of information and knowledge which needs to be effectively summarized to be useful. In this review, the main approaches to automatic text summarization are described. We review the different processes for summarization and describe the effectiveness and shortcomings of the different methods. Two types will be used i.e.-extractive approach and abstractive approach. The basic idea behind summarization is finding the subset of the data which contains the information of all the set. There is a great need to reduce unnecessary data. It is very difficult to summarize the document manually so there is the great need of automatic methods. Approaches have been proposed inspired by the application of deep learning methods for automatic machine translation, specifically by framing the problem of text summarization as a sequence-to-sequence learning problem.

# CONTENTS

[Abstract… …………………………………………………………………………..…..2](#_TOC_250001)

[Contents… ………………………………………………………………………….......3](#_TOC_250000)

1. Introduction… ……………………………………………………………….…….4-5
   1. Project Description………………………………………………………….......4
   2. About Natural Language Processing… ……………………………….....……...4
   3. Hardware Requirements…………………………………………………….…..5
   4. Software Requirements… …………………………………………………….…5
   5. Functional Requirements …………………………………………………….….5
2. Problem Definition… ……………………………………………………………….6
3. Objective……………………………………………………………………………6
4. Methodology……………………………………………………………………...7-9
5. Implementation Detail… ……………………………………………………….10-13
6. Future Scope… ……………………………………………………………………14
7. Conclusion…………………………………………………………………………15
8. References…………………………………………………………………………16

**INTRODUCTION**

## Project Description

With the developing measure of data, it has turned out to be hard to discover brief data. In this way, it is critical to making a framework that could condense like a human. Programmed content rundown with the assistance of Normal Dialect Handling is an instrument that gives synopses of a given archive. Content Outline strategies is divided in two ways i.e. - extractive and abstractive approach. The extractive approach basically choose the various and unique sentences, sections and so forth make a shorter type of the first report. The sentences are estimated and chosen based on accurate highlights of the sentences. In the Extractive technique, we have to choose the subset from the given expression or sentences in given frame of the synopsis. The extractive outline frameworks depends on two methods i.e. - extraction and expectation which includes the arrangement of the particular sentences that are essential in the general comprehension the archive. What’s more, the other methodology i.e. abstractive content synopsis includes producing completely new articulations to catch the importance of the first record. This methodology is all the more difficult but on the other hand is the methodology utilized by people.

**About Natural Language Processing**

Natural Language Processing (NLP) is the intersection of Computer Science, Linguistics and Machine Learning that is involved with the interaction between computers and humans in natural language. The NLTK will utilize a colossal instrument area and will make some help for individuals with the whole basic dialect taking care of system. This will assist individuals with “part sentences from sections, to part up words, seeing the syntactic segments of those words, denoting the fundamental subjects, doing this it serves to your machine by acknowledging the main thing to the substance.

## Requirements:

**Hardware Requirements:**

* Cloud
* Computer with GPU Capabilities

## Software Requirements:

## Python

## Functional Requirements:

## Natural Language Processing

## Python Library using Ntlk

**PROBLEM DEFINITION**

In the new period, where tremendous measure of data is accessible on the Web, it is most vital to give the enhanced gadget to get data rapidly. It is extremely intense for individuals to physically pick the synopsis of expansive archives of content. So there is an issue of scanning for vital reports from the accessible archives and discovering essential data. Along these lines programmed content rundown is the need of great importance. Content rundown is the way toward recognizing the most vital important data in a record or set of related archives. What’s more, compact them into a shorter rendition looking after its implications.

# OBJECTIVES

The objective of the project is to understand the concepts of natural language processing and creating a tool for text summarization. The concern in automatic summarization is increasing broadly so the manual work is removed. The project concentrates on creating a tool which automatically summarizes the document.

# METHODOLOGY

# Extractive Text Summarizer

# The Extractive summaries are used to highlight the words which are relevant, from input source document. Summaries help in generating concatenated sentences taken as per the appearance. Decision is made based on every sentence if that particular sentence will be included in the summary or not. For example, Search engines typically use Extractive summary generation methods to generate summaries from web page. Many types of logical and mathematical formulations have been used to create summary. The regions are scored and the words containing highest score are taken into the consideration. In extraction only important sentences are selected. This approach is easier to implement. There are three main obstacles for extractive approach. The first thing is ranking problem which includes ranking of the word. The second one selection problem that includes the selection of subset of particular units of ranks and the third one is coherence that is to know to select various units from understandable summary. There are many algorithms which are used to solve ranking problem. The two obstacles i.e. - selection and coherence are further solved to improve diversity and helps in minimizing the redundancy and pickup the lines which are important. Each sentence is scored and arranged in decreasing order according to the score. It is not trivial problem which helps in selecting the subsets of sentences for coherent summary. It helps in reduction of redundancy. When the list is put in ordered manner than the first sentence is the most important sentence which helps in forming the summary. The sentence having the highest similarity is selected in next step is picked from the top half of the list. The process has to be repeated until the limit is reached and relevant summary is generated

**Text Summarizer Steps:**

**Step 1: Collect data through web scrapper**

## The first library that we need to download is the [beautiful soup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) which is very useful Python utility for web scraping. Execute the following command at the command prompt to download the Beautiful Soup utility. Another important library that we need to parse XML and HTML is the [lxml](https://lxml.de/) library.  Import the important libraries required for scraping the data from the web. We then use the urlopen function from the urllib.request utility to scrape the data. Next, we need to call read function on the object returned by urlopen function in order to read the data. To parse the data, we use BeautifulSoup object and pass it the scraped data object i.e. article and the lxml parser.

## To retrieve the text we need to call find\_all function on the object returned by the BeautifulSoup. The tag name is passed as a parameter to the function. The find\_all function returns all the paragraphs in the article in the form of a list. All the paragraphs have been combined to recreate the article.

**Step 2: Cleanup data**

## The article\_text object contains text without brackets. However, we do not want to remove anything else from the article since this is the original article. We will not remove other numbers, punctuation marks and special characters from this text since we will use this text to create summaries and weighted word frequencies will be replaced in this article. Now we have two objects article\_text, which contains the original article and formatted\_article\_text which contains the formatted article. We will use formatted\_article\_text to create weighted frequency histograms for the words and will replace these weighted frequencies with the words in the article\_text object.

**Step3: Algorithm nltk to build tokens (words or sentences)**

## We will use the article\_text object for tokenizing the article to sentence since it contains full stops. The formatted\_article\_text does not contain any punctuation and therefore cannot be converted into sentences using the full stop as a parameter.

**Step4: Word frequency and Weighted word frequency for each word**

## To find the frequency of occurrence of each word, we use the formatted\_article\_text variable. We used this variable to find the frequency of occurrence since it doesn't contain punctuation, digits, or other special characters. we first store all the English stop words from the nltk library into a stopwords variable. Next, we loop through all the sentences and then corresponding words to first check if they are stop words. If not, we proceed to check whether the words exist in word\_frequency dictionary i.e. word\_frequencies, or not. If the word is encountered for the first time, it is added to the dictionary as a key and its value is set to 1. Otherwise, if the word previously exists in the dictionary, its value is simply updated by 1.

**Step 5: Calculate score for each sentences**

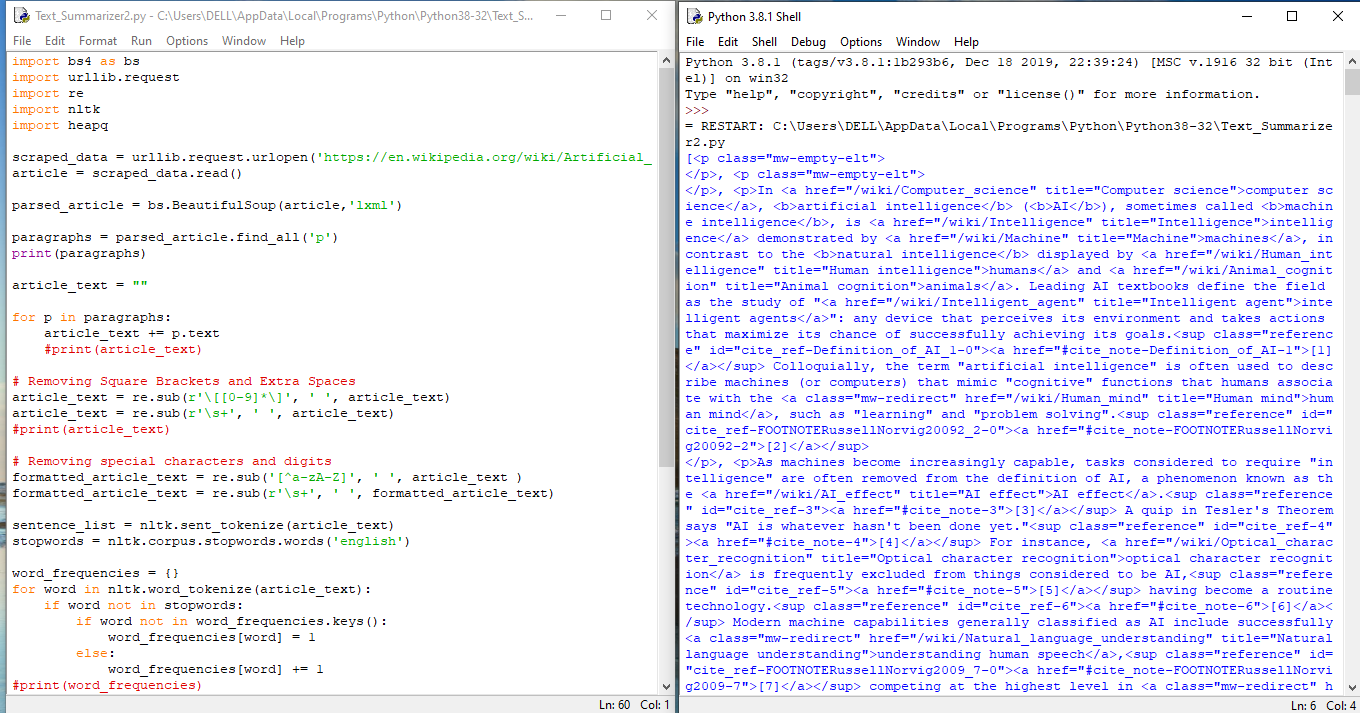
Now is the time to calculate the scores for each sentence by adding weighted frequencies of the words that occur in that particular sentence. we first create an empty sentence\_scores dictionary. The keys of this dictionary will be the sentences themselves and the values will be the corresponding scores of the sentences. Next, we loop through each sentence in the sentence\_list and tokenize the sentence into words. We then check if the word exists in the word\_frequencies dictionary. This check is performed since we created the sentence\_list list from the article\_text object; on the other hand, the word frequencies were calculated using the formatted\_article\_text object, which doesn't contain any stop words, numbers, etc.We do not want very long sentences in the summary, therefore, we calculate the score for only sentences with less than 30 words (although you can tweak this parameter for your own use-case). Next, we check whether the sentence exists in the sentence\_scores dictionary or not. If the sentence doesn't exist, we add it to the sentence\_scores dictionary as a key and assign it the weighted frequency of the first word in the sentence, as its value. On the contrary, if the sentence exists in the dictionary, we simply add the weighted frequency of the word to the existing value.

**Step 6: Select top 10 sentences for summary**

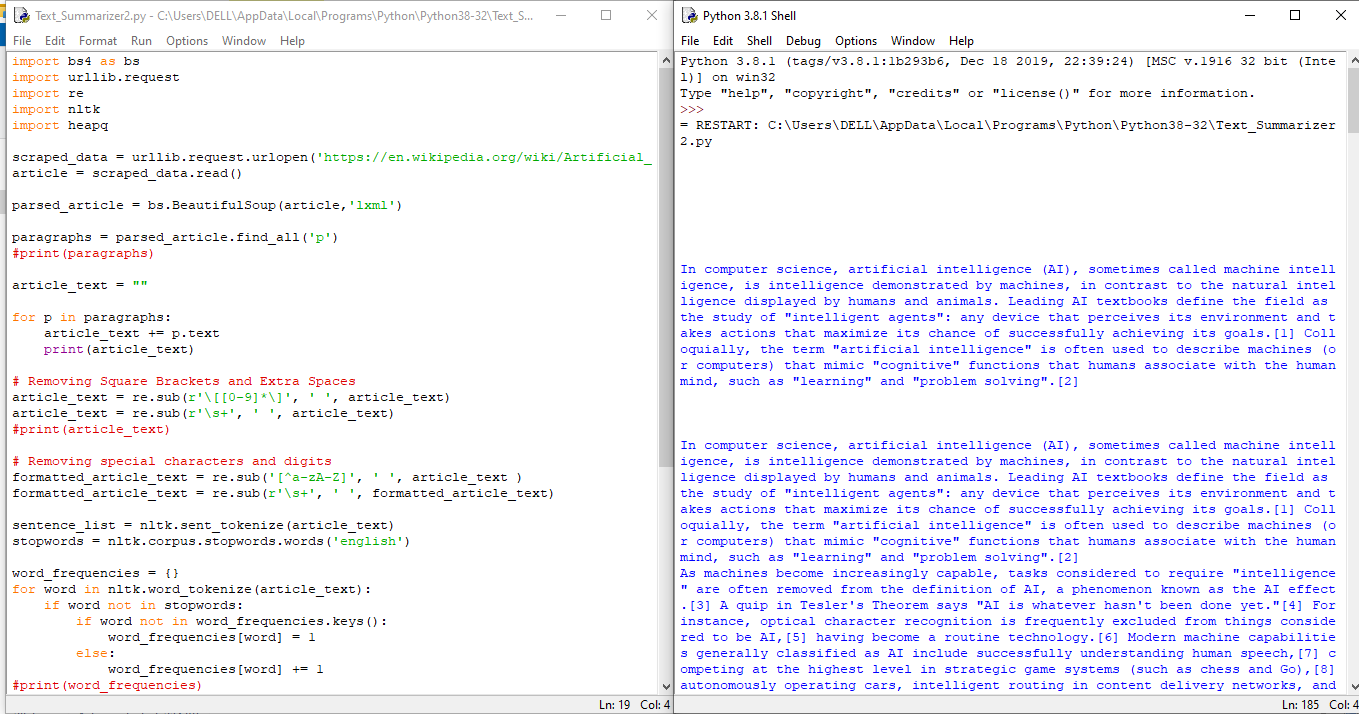
## We have the sentence\_scores dictionary that contains sentences with their corresponding score. To summarize the article, we can take top N sentences with the highest scores. The following script retrieves top 10 sentences and prints them on the screen. we use the heapq library and call its nlargest function to retrieve the top 10 sentences with the highest score

# IMPLEMENTATION DETAILS

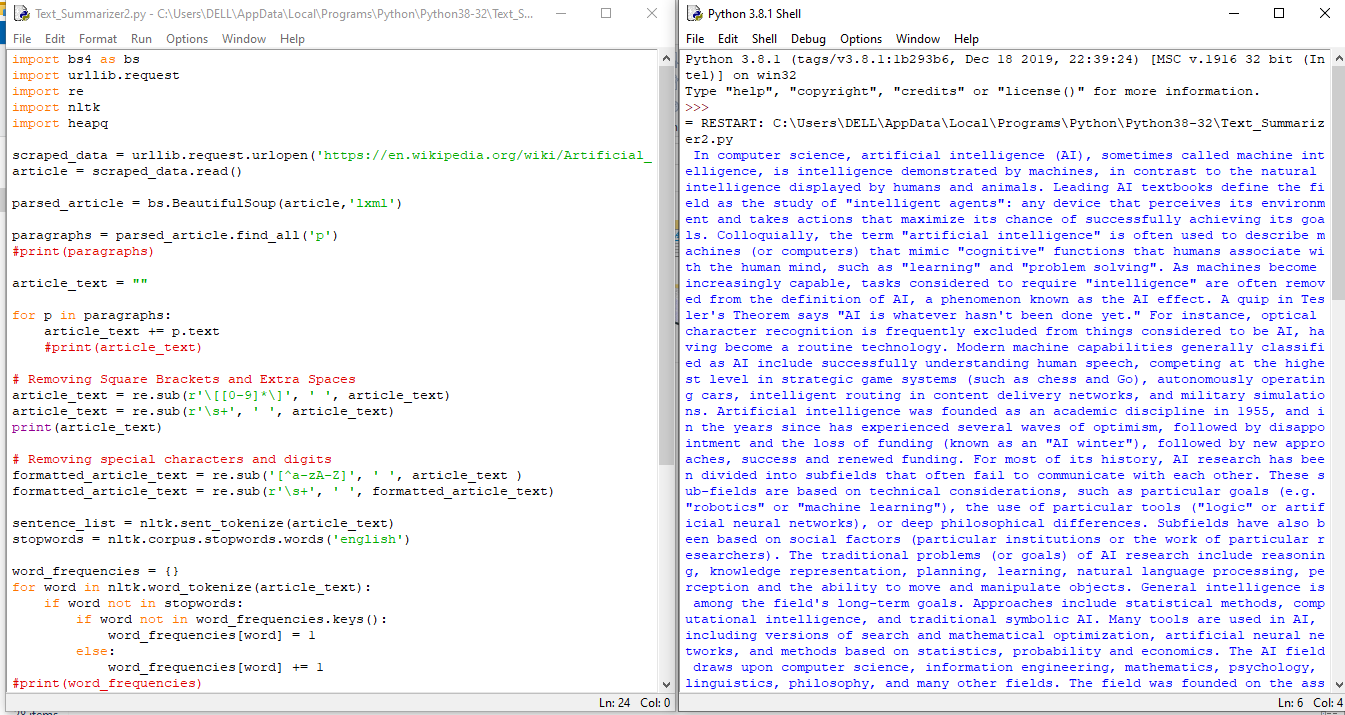
Screenshot of the implemented work



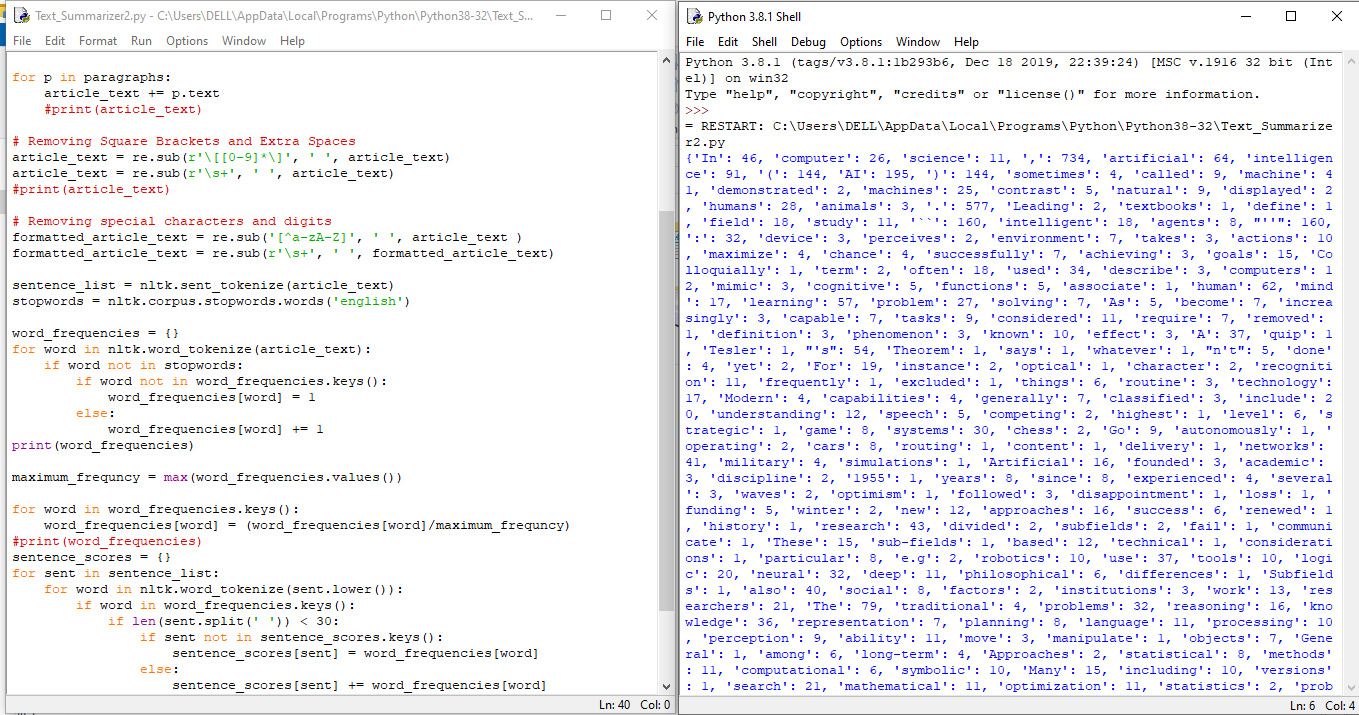
SS-1



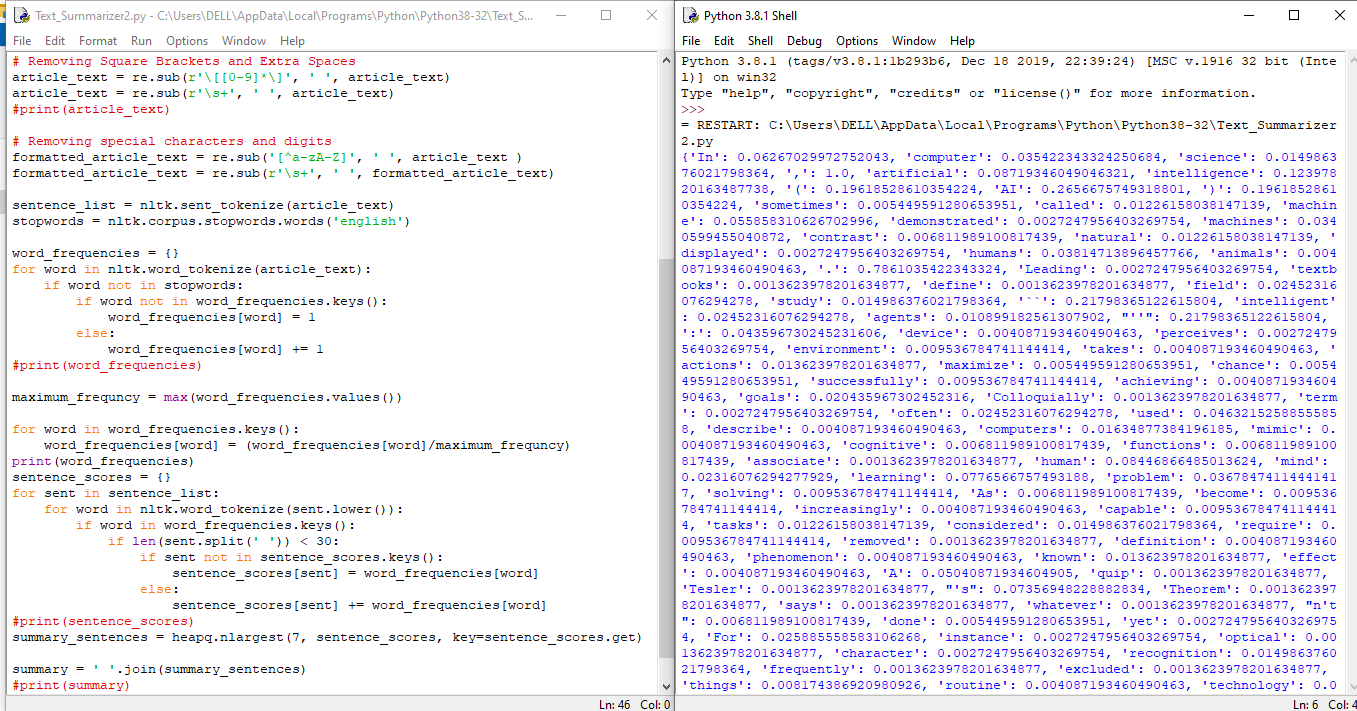
SS-2



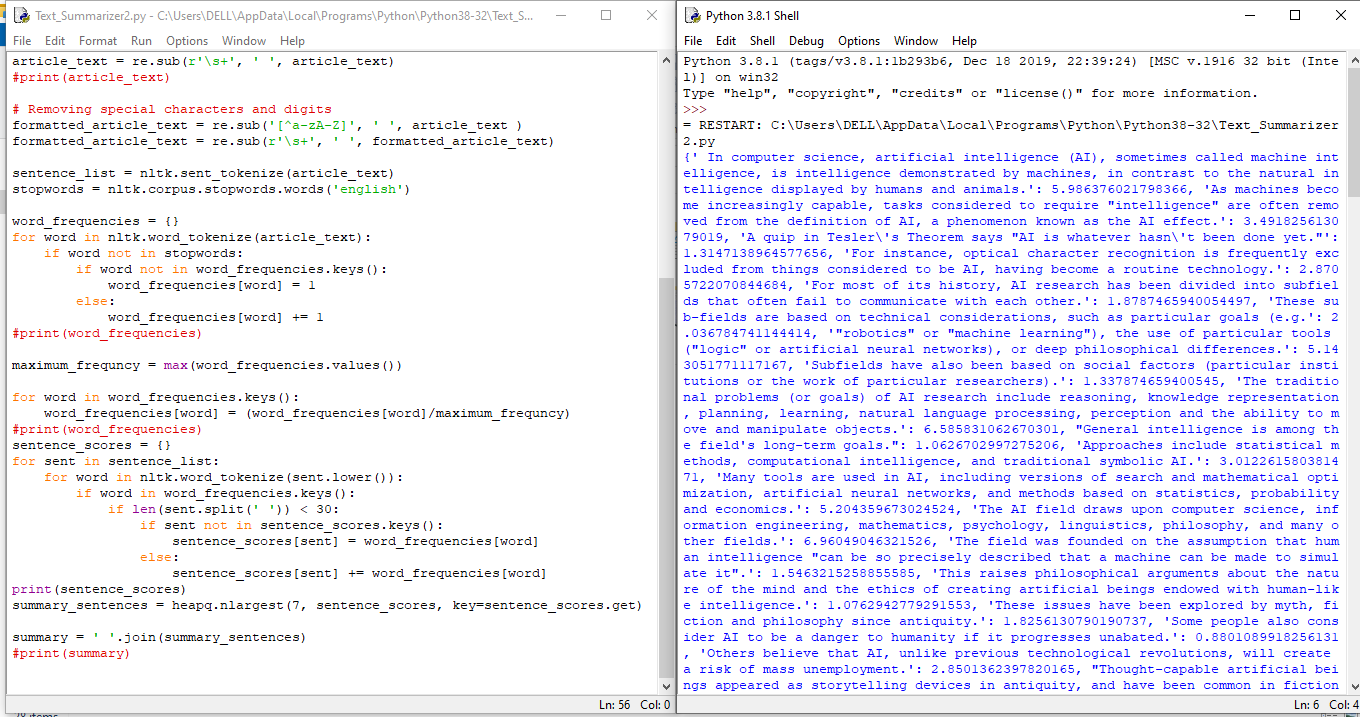
SS-3



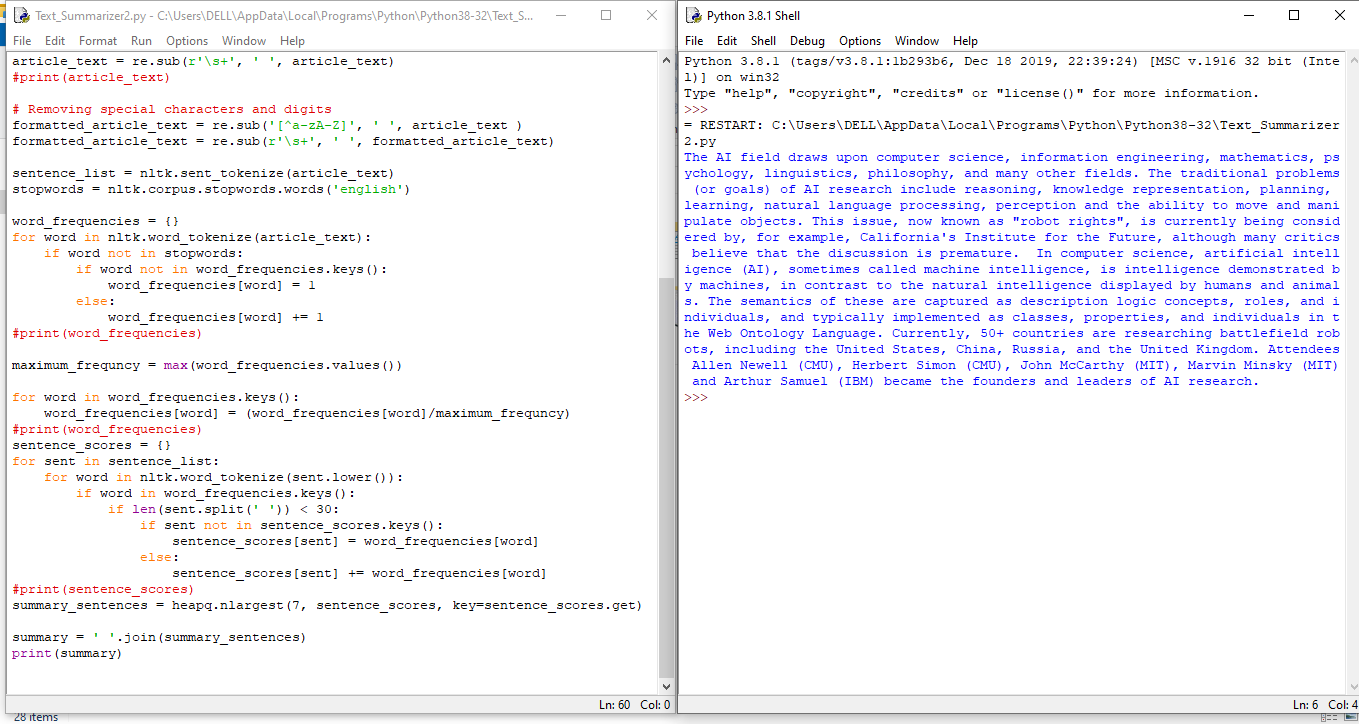
SS-4



SS-5



SS-6



SS-7

**FUTURE PROSPECTS**

The project is wide in scope all of the limitations stated below may seem to contradict that, but they are the only restrictions applied. This project looks at single document summarization the area of multi document summarization is not covered. Also, the summaries produced are largely extracts of the document being summarized, rather than newly generated abstracts. The parameters used are optimal for news articles, although that can be changed easily. With regard to language simplification, only lexical changes were considered syntactic changes were not. Background information was limited to biographical information and maps. We have implemented Automatic text summarization using abstractive method. Further, after using RNN and LSTM the accuracy is still very low for summarizer. Furthermore, we will be using machine learning for semantic text summarization for more accurate summaries and will try to make a grader which will grade the document according to English grammar. There are many text summarizers available but all does not give appropriate result. Thus we will be using machine learning algorithm to increase the effectiveness of the automatic summarizer.

**CONCLUSION**

There are approx 1,907,223,370 websites are active on the internet and approx 2,722,460 emails are being sent per second. This is an unbelievably huge amount of data. It is impossible for a user to get insights from such huge volumes of data. Furthermore, a large portion of this data is either redundant or doesn't contain much useful information. The most efficient way to get access to the most important parts of the data, without having to sift through redundant and insignificant data, is to summarize the data in a way that it contains non-redundant and useful information only. The data can be in any form such as audio, video, images, and text. Text summarization is a sub domain of [Natural Language Processing](https://stackabuse.com/what-is-natural-language-processing/) (NLP) that deals with extracting summaries from huge chunks of the text and can been proven helpful. Furthermore, even a website can be created.

**REFRENCES**

* https://www.researchgate.net/publication/315667326 Extractive Based Automatic Text Summarization
* <https://thetokenizer.com/2013/04/28/build-your-own-summary-tool/>
* <https://glowingpython.blogspot.in/2014/09/text-summarization-with-nltk.html>
* http://www.nltk.org/ NLTK 3.2.5 documentation