**Text Summarization using Natural Language Processing**

**Introduction**- Every day, we are inundated with information. There are numerous articles that we read on a daily basis. As a result, there is a lot of data moving about, largely in the form of text. If we need to learn something about an article, we must read the entire piece to understand it, and many times those articles become excessively long, such as a 5000-word article, which takes a long time. So, in order to receive the useful information contained in 1000 words, we must read the entire 5000-word article, which is a complete waste of time, and if we need to read several articles like that for work purposes, it will take a long time, resulting in a loss of work hour. The goal of text summarizing is to see if we can come up with a method that employs natural language processing to do so. This method will not only save time in comprehending a text, but it will also allow someone to read multiple texts in a short period of time, saving time in the long term.

**Objective-**

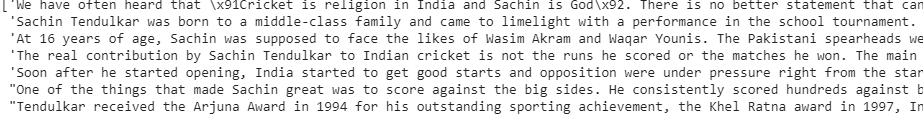
1. Extraction of useful information out of a huge amount of text.
2. Reduction of reading time.
3. Enable to read more articles as the time for reading each article will be reduced thus gather more information from different articles without losing much time.
4. Selecting articles which allows one to process more information when reading because only the most significant aspects of the content are captured.

**Problem Statement-**An article about Sachin Tendulkar has been collected from the internet which is made of around 691 words. Text summarization will be achieved using Natural Language Processing (NLP) to get important points about that article which are enough in gaining an understanding of the idea of the text.

The codes to achieve this text summarization is written below.

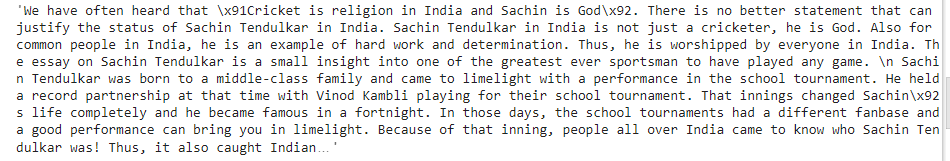
Google drive was mounted and the .txt file which contains the document was read and stored in a list named contents.

# google drive was mounted to read file  
from google.colab import drive  
drive.mount('/content/drive/')  
# file was read and stored in a list named contents  
f=open('/content/drive/MyDrive/Text.txt','r',encoding='latin1')  
f1=f.readlines()  
contents=[]  
for line in f1:  
contents.append(line)  
contents



The list was converted to string and then Unicode characters “\x91” and “]x92” were removed from the string and kept in a variable named text.

#list contents was converted to string and stored in text  
text = ' '.join([str(elem) for elem in contents])  
# Unicode character \x91 and \x92 was replaced with “‘” and kept in variable text  
text=text.replace("\x91","'")  
text=text.replace("\x92","'")  
text



The length of the text is found out.

len(text)



Number of words in the string

f=len(text.split())  
print ("The number of words in the given text is : " + str(f))



Importing the important libraries.

The spacy library is imported. From Spacy STOP\_WORDS have been imported.

From String, class punctuation has been imported

import spacy  
from spacy.lang.en.stop\_words import STOP\_WORDS  
from string import punctuation

A small size model "en\_core\_web\_sm" has been loaded.

nlp= spacy.load("en\_core\_web\_sm")

The whole text has been applied to nlp model and assigned to some doc object.

doc=nlp(text)

Iterate over every single token using list comprehension and these are the tokens to be worked upon.

tokens=[token.text for token in doc]

print(tokens)



These are the all punctuations and one extra punctuation '\n' has been added.

punctuation=punctuation+’\n’



Text Cleaning

An empty dictionary word\_freq has been created.

word\_freq={}

List of STOP\_WORDS has been stored in the stop\_words variable.

stop\_words= list(STOP\_WORDS)

A loop has been run over the doc to get those words that are not in the list of STOP\_WORDS and also not in the list of punctuations, and then the words were added to the word\_freq dictionary and the number of times they appear in doc has been added as a value in the dictionary.

for word in doc:  
 if word.text.lower() not in stop\_words:  
 if word.text.lower() not in punctuation:  
 if word.text not in word\_freq.keys():  
 word\_freq[word.text]= 1  
 else:  
 word\_freq[word.text]+= 1   
 print(word\_freq)

The maximum no of times a word appear has been figured out stored in variable max\_freq.

x=(word\_freq.values())

a=list(x)

a.sort()  
max\_freq=a[-1]  
max\_freq



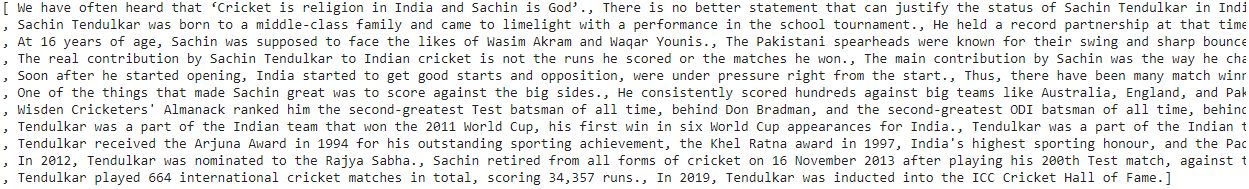
All the score of the words in word\_freq dictionary has been normalized by dividing each value in the dictionary by max\_freq and to do this a loop has been run on word\_freq dictionary and all the values were normalized.  
Sentence Tokenization

for word in word\_freq.keys():  
 word\_freq[word]=word\_freq[word]/max\_freq print(word\_freq)



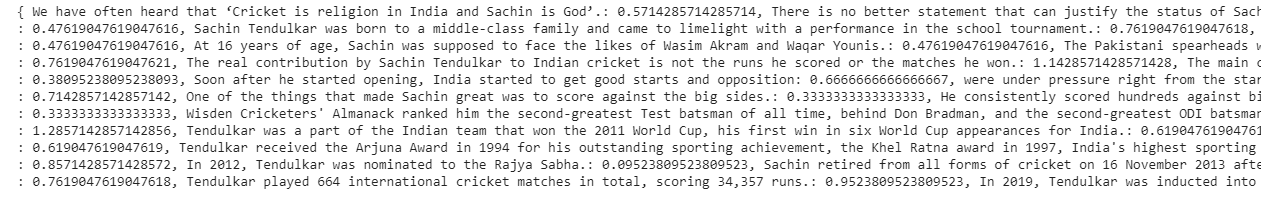
Sentences in doc object have been segmented by using list comprehension method and kept in variable sent\_tokens.

sent\_score={}  
sent\_tokens=[sent for sent in doc.sents]  
print(sent\_tokens)



Score of each individual sentence has been found out based on the word\_freq counter. An empty dictionary sent\_score has been created which will hold each sentence as a key and its value as a score. A loop was iterated on each individual sentence and it was checked the words in those sentences if appear in word\_freq dictionary and then based on the score of a word in word\_freq dictionary sent\_score has been determined.

for sent in sent\_tokens:  
 for word in sent:  
 if word.text.lower() in word\_freq.keys():  
 if sent not in sent\_score.keys():  
 sent\_score[sent]=word\_freq[word.text.lower()]  
 else:  
 sent\_score[sent]+= word\_freq[word.text.lower()]   
print(sent\_score)



Select 30% sentences with a maximum score

30% of sentences which is having maximum score out of this sent\_score dictionary have been grabbed.  
From heapq module nlargest library was imported.from total sent\_score 30% has been evaluated which comes to 8, which means maximum 13 sentences can be extracted which contains all important information.

from heapq import nlargest  
len(sent\_score) \*0.3

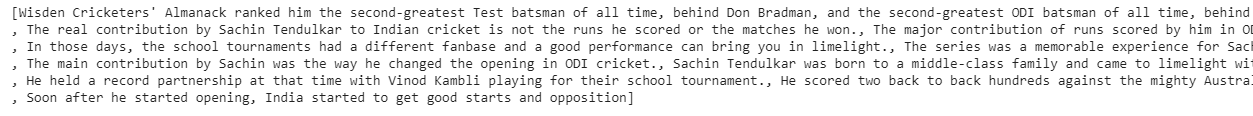


Getting Summary

Three parameters were passed to nlargest() function. First parameter is maximum number of sentense which in this case is 8. Second parameter is iterable on which we are going to apply this and in this case it's sent\_score. Third parameter is based on which key we are going to do all these things are here it s sent\_score.get,here get is used as a function which will return us those values sent\_score based on which we will get 8 sentences having 40% of maximum value.

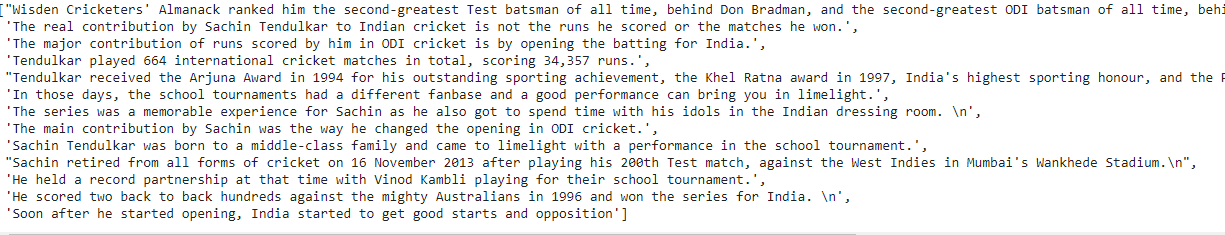
summary=nlargest(n=13,iterable=sent\_score,key=sent\_score.get)

print(summary)



List comprehension was applied to get the final summarized text.

final\_summary=[word.text for word in summary]  
final\_summary



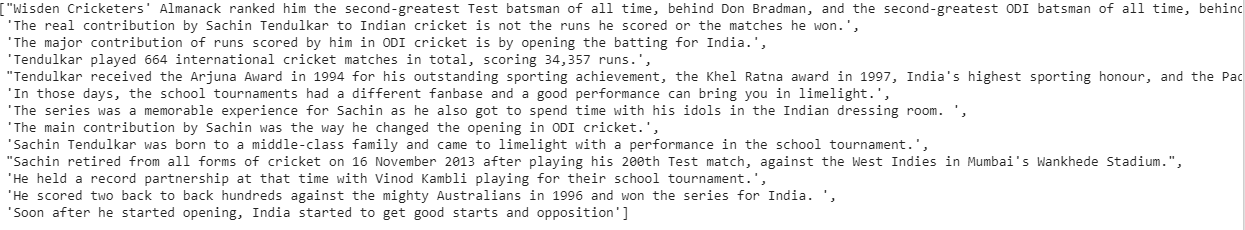
re module was imported to perform regex operation.

import re

Empty list f1 was created and a loop was run on the final extracted text, then regex operation was done to remove '\n' from all text and appended to list f1.

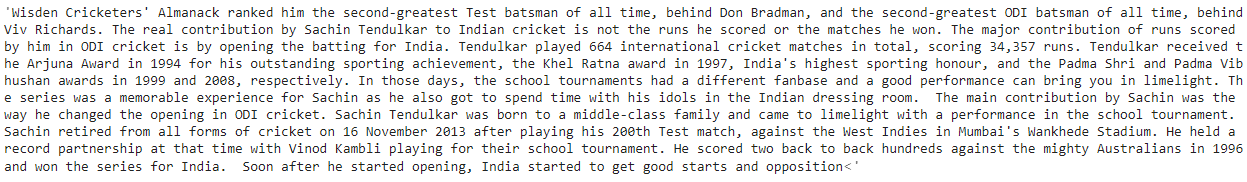
f1=[]  
for sub in final\_summary:  
 f1.append(re.sub('\n','',sub))

f1



List of final summarized text was converted to string using join() function and kept in variable f2.

f2=" ".join(f1)  
f2



The split() function was used to count the number of words in the final string.

f3=len(f2.split())  
print ("The number of words in final summary is : " + str(f3))



**Conclusion-** The article on Sachin Tendulkar was condensed into a 259-word document from a 691-word original, and this condensed document contains vital information that is the essence of the entire piece, making it understandable in a short amount of time.

My Linkedin ID is: [Linkedin](https://www.linkedin.com/in/ananda-chatterjee/)