

CSE-3024 Web Mining

Digital Assignment 1

Alokam Nikhitha

19BCE2555

CSE-3024 Web Mining

Lab Assignment 1

Alokam Nikhitha

19BCE2555

Question 1

Problem statement:

1. Write a python program to remove the stopwords for any given paragraph.

Create a set of stop words given below and print the output

```
stop_words = ['.',',','a','they','the','his','so','and','were','from','that','of','in','only','with','to']
```

Procedure:

- At First, we import the text file in our work space. To do this we can use open method of python which reads the file into our workspace.
- Next, we read each word into a variable as string using a nested for loop wherein we split each word whenever we encounter a space.
- Next, using regex in python we remove the punctuations from our string input. This will make sure that tokens are free from sentence structure.
- We tokenize each token in our text using split() function of NumPy lists and save it in a list.
- Then we use NumPy's unique method to only include unique tokens from our identified set of tokens.
- A tentative list of stop words and using a nested for loop we check if the given token belongs to that list or not, to remove stopwords. If it doesn't then we save it else we discard it.
- Finally, we print our list that contains the resultant tokens after removal of stop words.

Code:

#Reading input from a text file and saving it as a string

```
text = ""  
with open('test_file.txt') as file:  
    for line in file:  
        for word in line.split():  
            text= text + " " + word
```

#Removing punctuations from our input file

```
import re  
text = re.sub(r'^\w\s]', "", text)  
text
```

#Printing each token

```
print(text.split())
```

#Printing unique tokens

```
import numpy as np  
print(np.unique(text.split()))
```

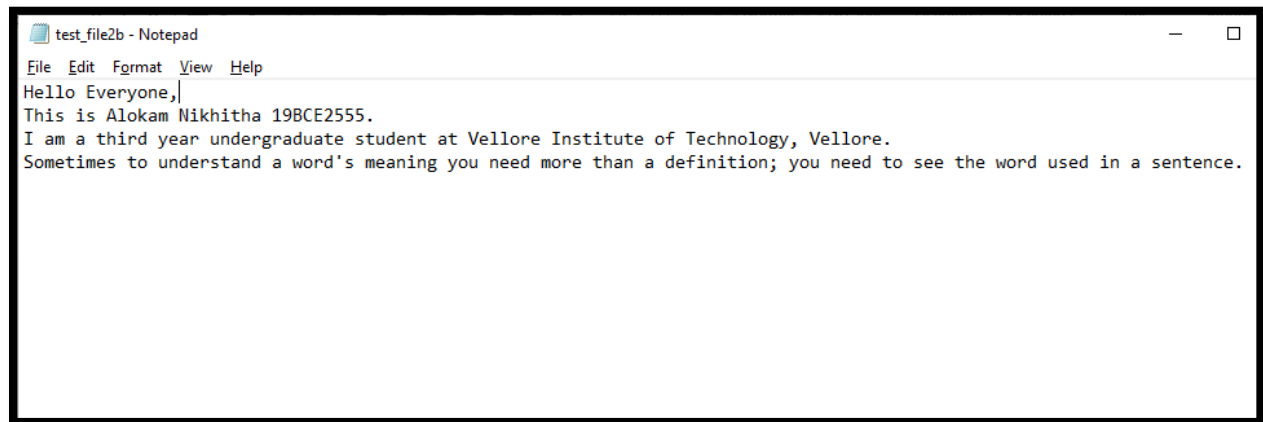
#Removing StopWords

```
stopwords = ["i", "a", "am", "and", "at", "for", "in", "is", "my", "of", "this"]  
res = []  
for x in tokens:  
    if x not in stopwords:  
        res.append(x)
```

#Printing cleaned tokens in our input text file

```
print(res)
```

Text File Taken as Input:



Code Snippets and Outputs:

```
In [3]: #Reading input from a text file and saving it as a string
text = ""
with open('test_file2b.txt') as file:
    for line in file:
        for word in line.split():
            text= text + " " + word.lower()
```

Here we are reading the text file using open method in python. Then reading each line we split each word and append it to a string variable with a space in between.

```
In [4]: #Removing punctuations from our input file
import re
text = re.sub(r'^\w\s]', '', text)
text
```

```
Out[4]: ' hello everyone this is alokam nikhitha 19bce2555 i am a third y
ear undergraduate student at vellore institute of technology vell
ore sometimes to understand a words meaning you need more than a
definition you need to see the word used in a sentence'
```

Here we are removing punctuations from our input file. This is done using regex, where we keep only alphanumeric inputs in our text string. We can see all the periods and commas from original input files are removed here.

```
In [5]: #Printing each token
tokens = text.split()
print(tokens)

['hello', 'everyone', 'this', 'is', 'alokam', 'nikhitha', '19bce2555', 'i', 'am', 'a', 'third', 'year', 'undergraduate', 'student', 'at', 'vellore', 'institute', 'of', 'technology', 'vellore', 'sometimes', 'to', 'understand', 'a', 'words', 'meaning', 'you', 'need', 'more', 'than', 'a', 'definition', 'you', 'need', 'to', 'see', 'the', 'word', 'used', 'in', 'a', 'sentence']
```

Next, we are splitting each word in our string using space character. Clearly, they form a token and hence we print each token.

```
In [6]: #Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)

['19bce2555' 'a' 'alokam' 'am' 'at' 'definition' 'everyone' 'hello' 'i' 'in' 'institute' 'is' 'meaning' 'more' 'need' 'nikhitha' 'of' 'see' 'sentence' 'sometimes' 'student' 'technology' 'than' 'the' 'third' 'this' 'to' 'undergraduate' 'understand' 'used' 'vellore' 'word' 'words' 'year' 'you']
```

Here we are only printing unique tokens from all the generated tokens using split method. This is done using NumPy's unique function, which identifies all the unique elements from a list.

```
In [7]: #Removing StopWords
stopwords = ["i", "a", "am", "and", "at", "for", "in", "is", "my", "of", "this"]
res = []
for x in tokens:
    if x not in stopwords:
        res.append(x)

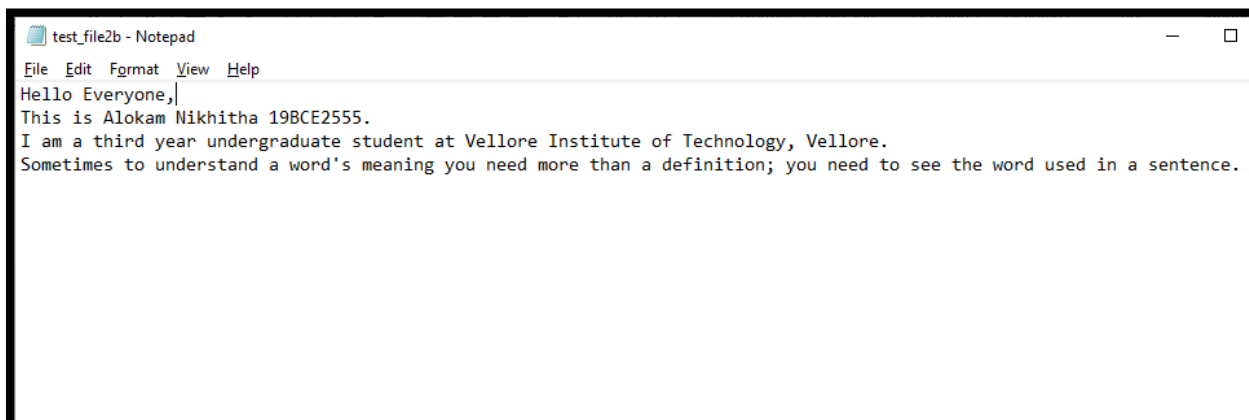
In [8]: #Printing cleaned tokens in our input text file
print(res)

['19bce2555', 'alokam', 'definition', 'everyone', 'hello', 'institute', 'meanin', 'more', 'need', 'nikhitha', 'see', 'sentence', 'sometimes', 'student', 'technology', 'than', 'the', 'third', 'to', 'undergraduate', 'understand', 'used', 'vellore', 'word', 'words', 'year', 'you']
```

Here we remove all the stop words from a self-defined list of stop words. We use nested loop to check if given token belongs to both tokens list and stopwords list. If it does, we don't add it to our result else we add it to our results.

Results and Output

- Input text:



```
test_file2b - Notepad
File Edit Format View Help
Hello Everyone,
This is Alokam Nikhitha 19BCE2555.
I am a third year undergraduate student at Vellore Institute of Technology, Vellore.
Sometimes to understand a word's meaning you need more than a definition; you need to see the word used in a sentence.
```

- Tokens of input text:

```
['19bce2555' 'a' 'alokam' 'am' 'at' 'definition' 'everyone' 'hello' 'i'  
'in' 'institute' 'is' 'meaning' 'more' 'need' 'nikhitha' 'of' 'see'  
'sentence' 'sometimes' 'student' 'technology' 'than' 'the' 'third' 'this'  
'to' 'undergraduate' 'understand' 'used' 'vellore' 'word' 'words' 'year'  
'you']
```

-Tokens after removal of stop words:

```
['19bce2555', 'alokam', 'definition', 'everyone', 'hello', 'institute', 'meanin  
g', 'more', 'need', 'nikhitha', 'see', 'sentence', 'sometimes', 'student', 'tec  
hnology', 'than', 'the', 'third', 'to', 'undergraduate', 'understand', 'used',  
'vellore', 'word', 'words', 'year', 'you']
```


Question 2

Problem statement:

2. Write a python program to tokenize
a) A sentence b) Multiple sentences (Without Nltk)

Procedure:

- Firstly, we import the text file in our work space. To do this we can use open method of python which reads the file into our workspace.
- Next, we read each word into a variable as string. This can be done using a nested for loop wherein we split each word whenever we encounter a space.
- Next, using regex in python we remove the punctuations from our string input. This will make sure that tokens are free from sentence structure.
- Finally, we will print each token and then unique tokens.

a)

Code:

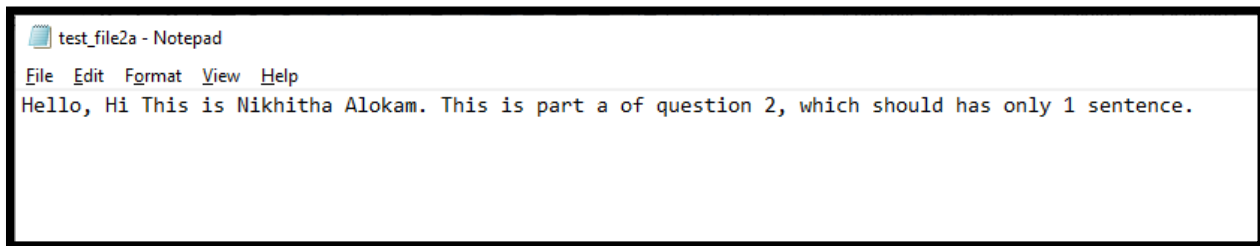
```
#Reading input from a text file and saving it as a string
text = ""
with open('test_file2a.txt') as file:
    for line in file:
        for word in line.split():
            text= text + " " + word
```

```
#Removing punctuations from our input file
import re
text = re.sub(r'^\w\s', "", text)
text
```

```
#Printing each token
print(text.split())
```

```
#Printing unique tokens
import numpy as np
print(np.unique(text.split()))
```

Text File Taken as Input:



Code Snippets and Outputs:

```
In [1]: #Reading input from a text file and saving it as a string
text = ""
with open('test_file2a.txt') as file:
    for line in file:
        for word in line.split():
            text= text + " " + word
```

Here we are reading the text file using open method in python. Later we are reading each line we split each word and append it to a string variable with a space in between.

```
In [2]: #Removing punctuations from our input file
import re
text = re.sub(r'^\w\s', '', text)
text
```

```
Out[2]: ' Hello Hi This is Nikhitha Alokam This is part a of question 2 which should has only 1 sentence'
```

Here we are removing punctuations from our input file using regex, where we keep only alphanumeric inputs in our text string. We can see all the periods and commas from original input files are removed here.

```
In [3]: #Printing each token
print(text.split())
```

```
['Hello', 'Hi', 'This', 'is', 'Nikhitha', 'Alokam', 'This', 'is', 'part', 'a', 'of', 'question', '2', 'which', 'should', 'has', 'only', '1', 'sentence']
```

Next, we are splitting each word in our string using space character. Clearly, they form a token and hence we print each token.

```
In [4]: import numpy as np
print(np.unique(text.split()))
```

```
['1' '2' 'Alokam' 'Hello' 'Hi' 'Nikhitha' 'This' 'a' 'has' 'is' 'of' 'only' 'part' 'question' 'sentence' 'should' 'which']
```

Here we are only printing unique tokens from all the generated tokens using split method. This is done using NumPy's unique function, which identifies all the unique elements from a list.

Results and Output

- Input text:

Hello, Hi This is Nikhitha Alokam. This is part a of question 2, which should has only 1 sentence.

- Tokens of input text:

```
['1' '2' 'Alokam' 'Hello' 'Hi' 'Nikhitha' 'This' 'a' 'has' 'is' 'of' 'only' 'part' 'question' 'sentence' 'should' 'which']
```

b)

Code:

```
#Reading input from a text file and saving it as a string
```

```
text = ""
```

```
with open('test_file2b.txt') as file:
```

```
    for line in file:
```

```
        for word in line.split():
```

```
            text= text + " " + word
```

```
#Removing punctuations from our input file
```

```
import re
```

```
text = re.sub(r'^\w\s', "", text)
```

```
text
```

```
#Printing each token
```

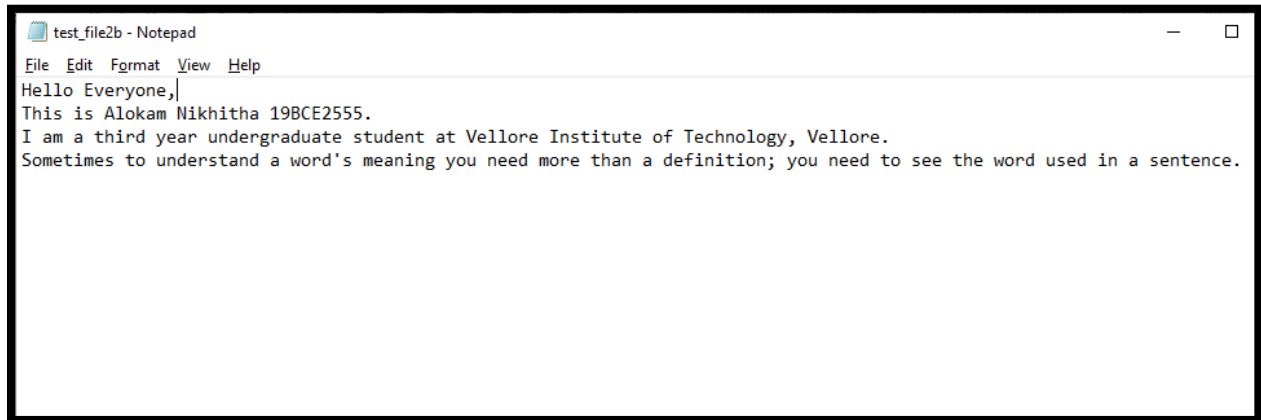
```
print(text.split())
```

```
#Printing unique tokens
```

```
import numpy as np
```

```
print(np.unique(text.split()))
```

Text File Taken as Input:



Code Snippets and Outputs:

```
In [5]: #Reading input from a text file and saving it as a string
text = ""
with open('test_file2b.txt') as file:
    for line in file:
        for word in line.split():
            text = text + " " + word
```

Here we are reading the text file using open method in python. Then reading each line we split each word and append it to a string variable with a space in between.

```
In [6]: #Removing punctuations from our input file
import re
text = re.sub(r'^\w\s', '', text)
text
```

```
Out[6]: ' Hello Everyone This is Alokam Nikhitha 19BCE2555 I am a third y
ear undergraduate student at Vellore Institute of Technology Vell
ore Sometimes to understand a words meaning you need more than a
definition you need to see the word used in a sentence'
```

Here we are removing punctuations from our input file. This is done using regex, where we keep only alphanumeric inputs in our text string. We can see all the periods and commas from original input files are removed here.

```
In [7]: #Printing each token
print(text.split())

['Hello', 'Everyone', 'This', 'is', 'Alokam', 'Nikhitha', '19BCE2555', 'I', 'am', 'a', 'third', 'year', 'undergraduate', 'student', 'at', 'Vellore', 'Institute', 'of', 'Technology', 'Vellore', 'Sometimes', 'to', 'understand', 'a', 'words', 'meaning', 'you', 'need', 'more', 'than', 'a', 'definition', 'you', 'need', 'to', 'see', 'the', 'word', 'used', 'in', 'a', 'sentence']
```

Next, we are splitting each word in our string using space character. Clearly, they form a token and hence we print each token.

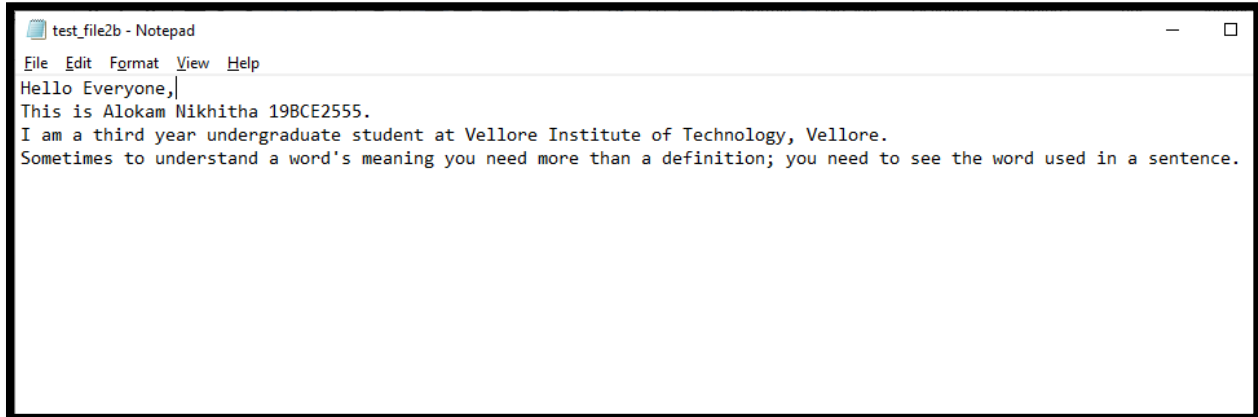
```
In [8]: import numpy as np
print(np.unique(text.split()))

['19BCE2555' 'Alokam' 'Everyone' 'Hello' 'I' 'Institute' 'Nikhitha'
 'Sometimes' 'Technology' 'This' 'Vellore' 'a' 'am' 'at' 'definition' 'in'
 'is' 'meaning' 'more' 'need' 'of' 'see' 'sentence' 'student' 'than' 'the'
 'third' 'to' 'undergraduate' 'understand' 'used' 'word' 'words' 'year'
 'you']
```

Here we are only printing unique tokens from all the generated tokens using split method. This is done using NumPy's unique function, which identifies all the unique elements from a list.

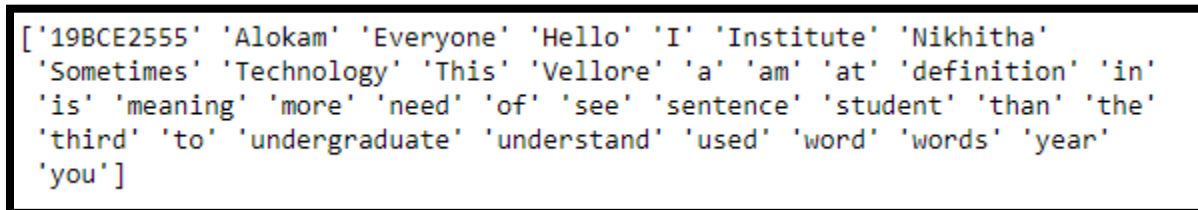
Results and Output:

- Input text:



```
test_file2b - Notepad
File Edit Format View Help
Hello Everyone,
This is Alokam Nikhitha 19BCE2555.
I am a third year undergraduate student at Vellore Institute of Technology, Vellore.
Sometimes to understand a word's meaning you need more than a definition; you need to see the word used in a sentence.
```

- Tokens of input text:



```
['19BCE2555' 'Alokam' 'Everyone' 'Hello' 'I' 'Institute' 'Nikhitha'
'Sometimes' 'Technology' 'This' 'Vellore' 'a' 'am' 'at' 'definition' 'in'
'is' 'meaning' 'more' 'need' 'of' 'see' 'sentence' 'student' 'than' 'the'
'third' 'to' 'undergraduate' 'understand' 'used' 'word' 'words' 'year'
'you']
```

CSE-3024 Web Mining

Lab Assignment 2

Alokam Nikhitha

19BCE2555

Question

Experiment-2 (21-01-2022)

1. Write a program (using nltk toolkit in python environment) to tokenize
 - a) Sentence
 - b) A paragraph

Problem statement:

To write a program to tokenize using nltk toolkit in Python Environment.

Procedure:

- We firstly import our text file into our workspace. To do that we are able to use open method of python which will read our text file to the workspace.
- Next, we will import the necessary NLTK libraries including stopwords, sent_tokenize and word_tokenize.
- Using word_tokenize we tokenize each word and store it in a variable list named tokens
- Then, we will read every word in input file as a string input. This may be carried out using nested for loop in which we split every word whenever we come across a space.
- Then , we use regex to remove the punctuations from the input string. This will render token a higher syntactic shape and break the sentence bonds.
- Then ,we split and store every token right into a list with nltk's sentence_tokenize method.
- Then subsequently we remove stop words in the same process as in preceding assignment.
- Finally, we print our list that contains the resultant tokens post removal of stop words as well.

a) Sentence

Code:

```
#Reading input (Single ) from a text file
text = ""
with open('test_file2a.txt') as file:
    for line in file:
        for word in line.split():
            text = text + " " + word.lower()

#Importing libraries
import re
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords

#Removing punctuations from our input .
text = re.sub(r'^\w\s', "", text)
text

#Printing each token
tokens = word_tokenize(text)
print(tokens)

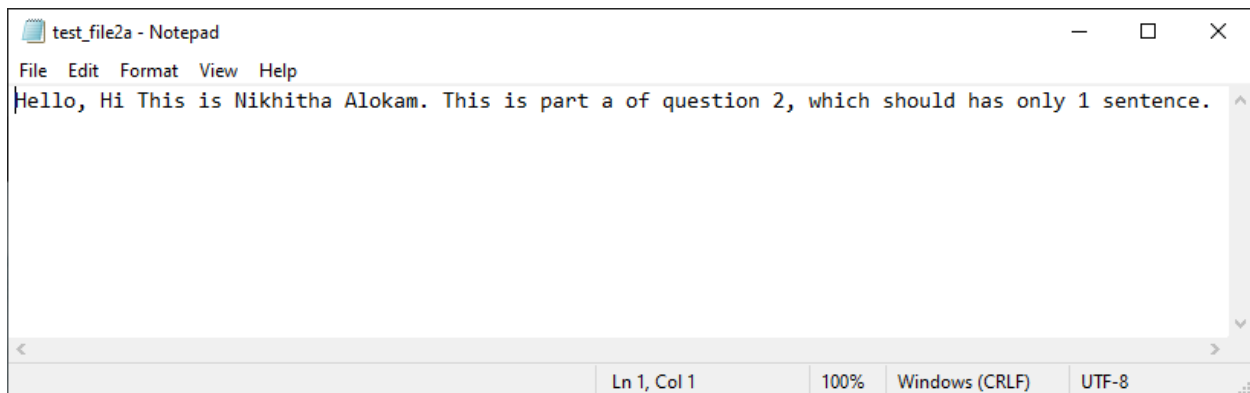
#Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)

#Removing Stopwords
res = []
for x in tokens:
    if x not in set(stopwords.words('english')):
```

```
res.append(x)
```

```
#Printing cleaned tokens in our input text file  
print(res)
```

Text File Taken as Input:



Code Snippets and Outputs:

```
In [1]: #Reading input (Single ) from a text file  
text = ""  
with open('test_file2a.txt') as file:  
    for line in file:  
        for word in line.split():  
            text = text + " " + word.lower()
```

Here we're analyzing the text file by using open approach in python. Then analyzing every line we split every word and append it to a string variable with a space in between.

```
In [2]: #Importing Libraries
import re
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
```

Here we're importing the necessary libraries which incorporates our library of choice this is nltk. We additionally import stopwords and word_tokenize, sentence_tokenize from nltk. To eliminate punctuations we import the regex library.

```
In [3]: #Removing punctuations from our input file
text = re.sub(r'^\w\s', '', text)
text
```

```
Out[3]: 'hello hi this is nikhitha alokam this is part a of question 2 which should ha
s only 1 sentence'
```

Here we're removing punctuations from the input taken from input file. This is carried out by using regex, wherein we keep only alphanumeric inputs in our text string. We can see all of the intervals and commas from original input file are eliminated here.

```
In [4]: #Printing each token
tokens = word_tokenize(text)
print(tokens)
```

```
['hello', 'hi', 'this', 'is', 'nikhitha', 'alokam', 'this', 'is', 'part', 'a',
'of', 'question', '2', 'which', 'should', 'has', 'only', '1', 'sentence']
```

Since we have to tokenize every word, we have used the word_tokenize and as we can see each token is identified here and we print them.

```
In [5]: #Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)

['1' '2' 'a' 'alokam' 'has' 'hello' 'hi' 'is' 'nikhitha' 'of' 'only'
 'part' 'question' 'sentence' 'should' 'this' 'which']
```

Here we print all the unique tokens in after tokenizing. And print them.

```
In [7]: #Removing Stopwords
res = []
for x in tokens:
    if x not in set(stopwords.words('english')):
        res.append(x)
```

```
In [8]: #Printing cleaned tokens in our input text file
print(res)

['1', '2', 'alokam', 'hello', 'hi', 'nikhitha', 'part', 'question', 'sentence']
```

Here we remove all the stop words from a self-defined list of stop words. We use nested loop to check if given token belongs to both tokens list and stopwords list. If it does, we don't add it to our result else we add it to our results.

Results and Output

1. Input Sentence

```
Hello, Hi This is Nikhitha Alokam. This is part a of question 2, which should has only 1 sentence.
```

2. Tokens with out removing Stopwords.

```
['1' '2' 'a' 'alokam' 'has' 'hello' 'hi' 'is' 'nikhitha' 'of' 'only'
 'part' 'question' 'sentence' 'should' 'this' 'which']
```

3. Tokens after removing Stopwords

```
['1', '2', 'alokam', 'hello', 'hi', 'nikhitha', 'part', 'question', 'sentence']
```

b) A Paragraph

Code:

```
#Reading input (Single ) from a text file
text = ""
with open('test_file2b.txt') as file:
    for line in file:
        for word in line.split():
            text = text + " " + word.lower()

#Importing libraries
import re
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords

#Removing punctuations from our input .
text = re.sub(r'^\w\s', "", text)
text

#Printing each token
tokens = word_tokenize(text)
print(tokens)

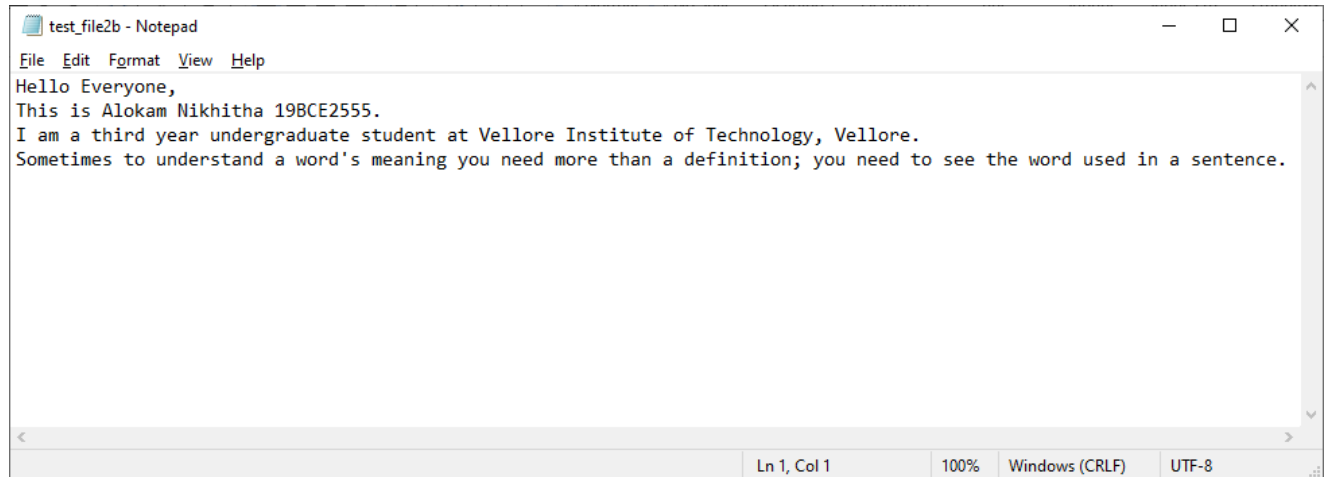
#Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)

#Removing Stopwords
res = []
for x in tokens:
    if x not in set(stopwords.words('english')):
```

```
res.append(x)
```

```
#Printing cleaned tokens in our input text file  
print(res)
```

Text File Taken as Input:



Code Snippets and Outputs:

```
In [9]: #Reading input (A paragraph) from a text file  
text = ""  
with open('test_file2b.txt') as file:  
    for line in file:  
        for word in line.split():  
            text = text + " " + word.lower()
```

Here we're analyzing the text file by using open approach in python. Then analyzing every line we split every word and append it to a string variable with a space in between.

```
In [10]: #Importing libraries
import re
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
```

Here we're importing the necessary libraries which incorporates our library of choice this is nltk. We additionally import stopwords and word_tokenize, sentence_tokenize from nltk. To eliminate punctuations we import the regex library

```
In [11]: #Removing punctuations from our input file
text = re.sub(r'^\w\s', '', text)
text
```

```
Out[11]: 'hello everyone this is alokam nikhitha 19bce2555 i am a third year undergradu
ate student at vellore institute of technology vellore sometimes to understand
a words meaning you need more than a definition you need to see the word used i
n a sentence'
```

Here we're removing punctuations from the input taken from input file. This is carried out by using regex, wherein we keep only alphanumeric inputs in our text string. We can see all of the intervals and commas from original input file are eliminated here.

```
In [12]: #Printing each token
tokens = word_tokenize(text)
print(tokens)

['hello', 'everyone', 'this', 'is', 'alokam', 'nikhitha', '19bce2555', 'i', 'a', 'm', 'a', 'third', 'year', 'undergraduate', 'student', 'at', 'vellore', 'institute', 'of', 'technology', 'vellore', 'sometimes', 'to', 'understand', 'a', 'words', 'meaning', 'you', 'need', 'more', 'than', 'a', 'definition', 'you', 'need', 'to', 'see', 'the', 'word', 'used', 'in', 'a', 'sentence']
```

Since we have to tokenize every word, we have used the word_tokenize and as we can see each token is identified here and we print them.


```
In [13]: #Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)

['19bce2555' 'a' 'alokam' 'am' 'at' 'definition' 'everyone' 'hello' 'i'
'in' 'institute' 'is' 'meaning' 'more' 'need' 'nikhitha' 'of' 'see'
'sentence' 'sometimes' 'student' 'technology' 'than' 'the' 'third' 'this'
'to' 'undergraduate' 'understand' 'used' 'vellore' 'word' 'words' 'year'
'you']
```

Here we print all the unique tokens in after tokenizing. And print them.

```
In [14]: #Removing Stopwords
res = []
for x in tokens:
    if x not in set(stopwords.words('english')):
        res.append(x)
```

```
In [15]: #Printing cleaned tokens in our input text file
print(res)

['19bce2555', 'alokam', 'definition', 'everyone', 'hello', 'institute', 'meanin
g', 'need', 'nikhitha', 'see', 'sentence', 'sometimes', 'student', 'technolog
y', 'third', 'undergraduate', 'understand', 'used', 'vellore', 'word', 'words',
'year']
```

Here we remove all the stop words from a self-defined list of stop words. We use nested loop to check if given token belongs to both tokens list and stopwords list. If it does, we don't add it to our result else we add it to our results.

Results and Output

1.Input Sentence

Hello Everyone,
This is Alokam Nikhitha 19BCE2555.
I am a third year undergraduate student at Vellore Institute of Technology, Vellore.
Sometimes to understand a word's meaning you need more than a definition; you need to see the word used in a sentence.

2.Tokens with out removing Stopwords.

```
['19bce2555' 'a' 'alokam' 'am' 'at' 'definition' 'everyone' 'hello' 'i'  
'in' 'institute' 'is' 'meaning' 'more' 'need' 'nikhitha' 'of' 'see'  
'sentence' 'sometimes' 'student' 'technology' 'than' 'the' 'third' 'this'  
'to' 'undergraduate' 'understand' 'used' 'vellore' 'word' 'words' 'year'  
'you']
```

3.Tokens after removing Stopwords

```
['19bce2555', 'alokam', 'definition', 'everyone', 'hello', 'institute', 'meanin  
g', 'need', 'nikhitha', 'see', 'sentence', 'sometimes', 'student', 'technolog  
y', 'third', 'undergraduate', 'understand', 'used', 'vellore', 'word', 'words',  
'year']
```

CSE-3024 Web Mining

Lab Assignment 3

Alokam Nikhitha

19BCE2555

Question

Experiment-3 (19-01-2022)

Write a python program to find the important words from the text using TF-IDF.

Use minimum of 5 documents with the real text source from a web page of some relevance.

Step by Step Implementation of the TF-IDF Model

Problem statement:

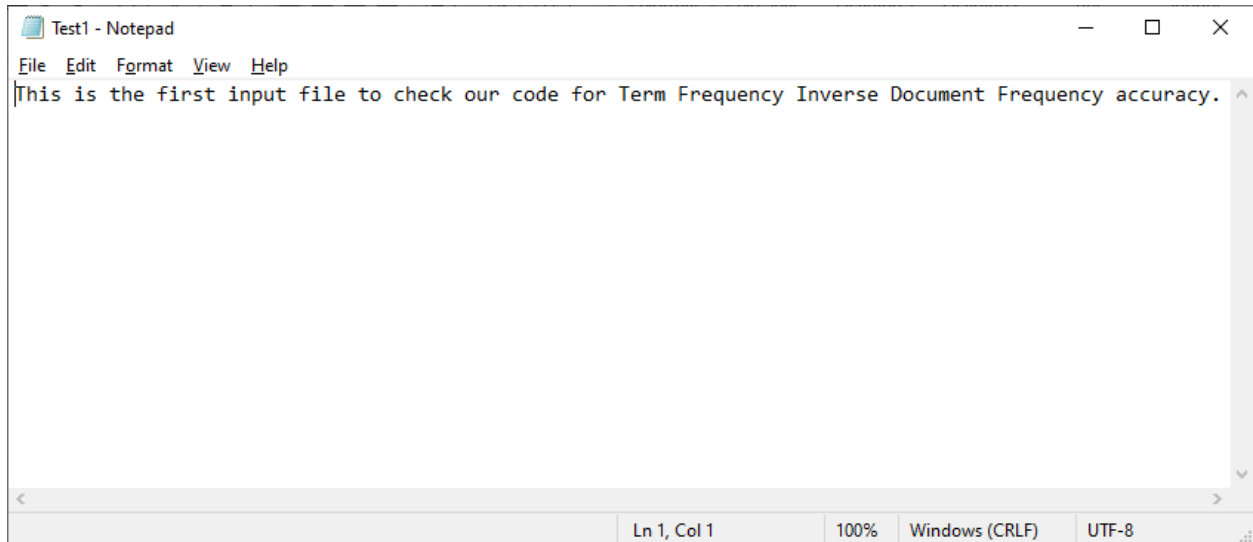
Python program to find the important words from the textfile using TF-IDF using atleast minimum of 5 documents

Procedure:

- We will Firstly import our libraries Which are required in doing the term frequency count.
- Later, we will declare and define tf, idf, n_containing and tf_idf functions that will help assist the return values and make code more readable.
- We will create 5 Text File inputs and read them in our workspace.
- Later, We will make the bloblist that contains all the Text File Inputs in list format. And then we will print the counts of top 3 words in every document.
- We will then calculate the cosine similarity using inbuilt cosine_similarity matrix.
- For the above we need to create a pandas data frame of count vectors.

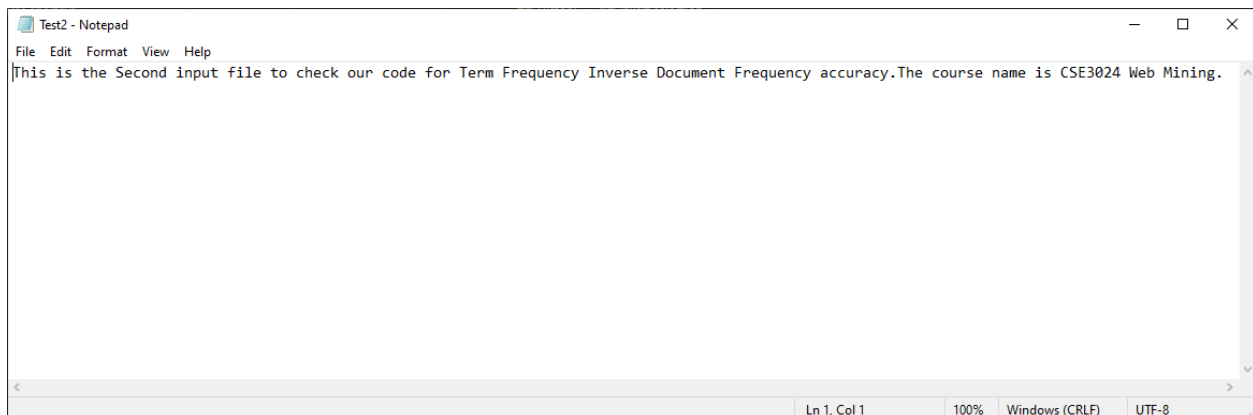
Text File Taken as Input:

Text File 1:



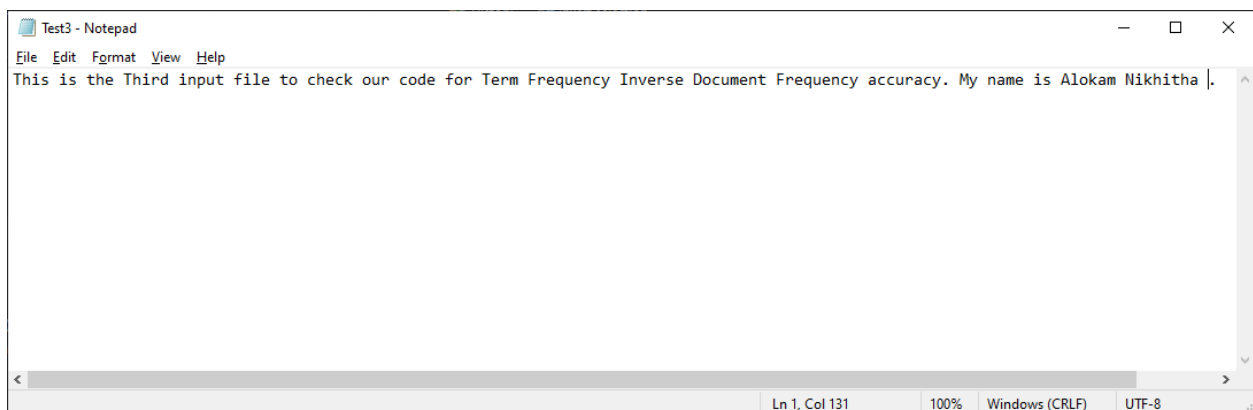
A screenshot of a Notepad window titled "Test1 - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains the sentence: "This is the first input file to check our code for Term Frequency Inverse Document Frequency accuracy." The status bar at the bottom shows "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

Text File 2:



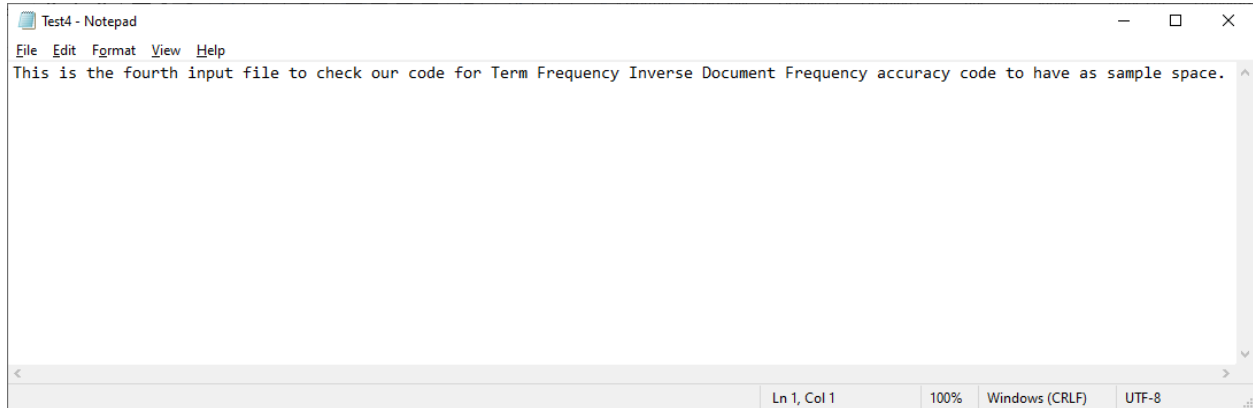
A screenshot of a Notepad window titled "Test2 - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains the sentence: "This is the Second input file to check our code for Term Frequency Inverse Document Frequency accuracy. The course name is CSE3024 Web Mining." The status bar at the bottom shows "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

Text File 3:

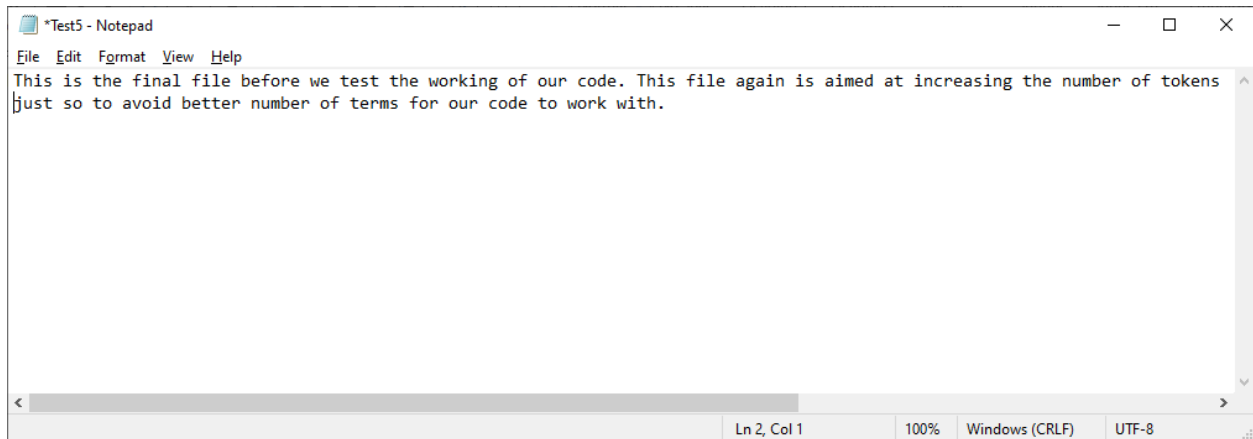


A screenshot of a Notepad window titled "Test3 - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains the sentence: "This is the Third input file to check our code for Term Frequency Inverse Document Frequency accuracy. My name is Alokam Nikhitha." The status bar at the bottom shows "Ln 1, Col 131", "100%", "Windows (CRLF)", and "UTF-8".

Text File 4:



Text File 5:



Code:

```
In [1]: #Importing the Libraries
import math
from textblob import TextBlob as tb
```

```
In [2]: #Creating the Term Frequency return function
def tf(word, blob):
    return blob.words.count(word)
```

```
In [3]: #Creaeting containing function
def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob.words)
```

```
In [4]: #Function to return Inverse Document Frequency
def idf(word, bloblist):
    return math.log(len(bloblist))/(1+n_containing(word, bloblist))
```

```
In [5]: #Function to return Term Frequency-Inverse Document Frequency
def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)
```

```
In [6]: #Reading First Input File
with open('Test1.txt') as a:
    test1 = (a.read())
document1 = tb(test1)
```

```
In [7]: #Reading Second Input File
with open('Test2.txt') as a:
    test2 = (a.read())
document2 = tb(test2)
```

```
In [8]: #Reading Third Input File
with open('Test3.txt') as a:
    test3 = (a.read())
document3 = tb(test3)
```

```
In [9]: #Reading Fourth Input File
with open('Test4.txt') as a:
    test4 = (a.read())
document4 = tb(test4)
```

```
In [10]: #Reading Fifth Input File
with open('Test5.txt') as a:
    test5 = (a.read())
document5 = tb(test5)
```

```
In [10]: #Reading Fifth Input File
with open('Test5.txt') as a:
    test5 = (a.read())
document5 = tb(test5)
```

```
In [11]: #Printing the top three words in each document
bloblist = [document1, document2, document3, document4, document5]
for i, blob in enumerate(bloblist):
    print("Top words in document {}".format(i+1))
    scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    for word, score in sorted_words[:3]:
        print("\tWord: {}, TF-IDF: {}".format(word, round(score, 5)))
```

```
In [12]: #Calculating Cosine Similarity
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
documents = [test1, test2, test3, test4, test5]
```

```
In [13]: #Creating the Document Term Matrix
count_vectorizer = CountVectorizer()
sparse_matrix = count_vectorizer.fit_transform(documents)
```

```
In [14]: #Creating a dataframe to store each count_vectorizer
doc_term_matrix = sparse_matrix.todense()
df = pd.DataFrame(doc_term_matrix,
                  columns=count_vectorizer.get_feature_names(),
                  index=['test1', 'test2', 'test3', 'test4', 'test5'])
df
```

```
In [15]: #Printing the Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df, df))
```


Code Snippets and Outputs:

```
In [1]: #Importing the Libraries
import math
from textblob import TextBlob as tb
```

Here we are importing the necessary Libraries

```
In [2]: #Creating the Term Frequency return function
def tf(word, blob):
    return blob.words.count(word)
```

Here we are creating the Term Frequency return Function which takes word and blob as attributes.

```
In [3]: #Creating containing function
def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob.words)
```

Here we are now creating the n_containing Function which takes words and bloblist as attributes.

```
In [4]: #Function to return Inverse Document Frequency
def idf(word, bloblist):
    return math.log(len(bloblist))/(1+n_containing(word, bloblist))
```

A function named idf is created in order to Inverse the Document Frequency

```
In [5]: #Function to return Term Frequency-Inverse Document Frequency
def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)
```

Here we create a Function named tfidf to return Term Frequency-Inverse Document Frequency

```
In [6]: #Reading First Input File
with open('Test1.txt') as a:
    test1 = (a.read())
document1 = tb(test1)
```

```
In [7]: #Reading Second Input File
with open('Test2.txt') as a:
    test2 = (a.read())
document2 = tb(test2)
```

```
In [8]: #Reading Third Input File
with open('Test3.txt') as a:
    test3 = (a.read())
document3 = tb(test3)
```

```
In [9]: #Reading Fourth Input File
with open('Test4.txt') as a:
    test4 = (a.read())
document4 = tb(test4)
```

```
In [10]: #Reading Fifth Input File
with open('Test5.txt') as a:
    test5 = (a.read())
document5 = tb(test5)
```

Here we are reading all the 5 Input Text Files(i.e, Test1.txt, Test2.txt, Test3.txt, Test4.txt, Test5.txt)

```
In [11]: #Printing the top three words in each document
bloblist = [document1, document2, document3, document4, document5]
for i, blob in enumerate(bloblist):
    print("Top words in document {}".format(i+1))
    scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    for word, score in sorted_words[:3]:
        print("\tWord: {}, TF-IDF: {}".format(word, round(score, 5)))
```

```
Top words in document 1
    Word: first, TF-IDF: 0.80472
    Word: Frequency, TF-IDF: 0.64378
    Word: accuracy, TF-IDF: 0.40236
Top words in document 2
    Word: Second, TF-IDF: 0.80472
    Word: accuracy.The, TF-IDF: 0.80472
    Word: course, TF-IDF: 0.80472
Top words in document 3
    Word: Third, TF-IDF: 0.80472
    Word: My, TF-IDF: 0.80472
    Word: Alokam, TF-IDF: 0.80472
Top words in document 4
    Word: fourth, TF-IDF: 0.80472
    Word: have, TF-IDF: 0.80472
    Word: as, TF-IDF: 0.80472
Top words in document 5
    Word: of, TF-IDF: 2.41416
    Word: number, TF-IDF: 1.60944
    Word: the, TF-IDF: 0.80472
```

Here we've printed the top words in every document. We've printed only top 3 words and the TF-IDF values of them in the same line with the word/term.

```
In [12]: #Calculating Cosine Similarity
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
documents = [test1, test2, test3, test4, test5]
```

Here we are Calculating the Cosine Similarity of all the Input Text files.

```
In [13]: #Creating the Document Term Matrix
count_vectorizer = CountVectorizer()
sparse_matrix = count_vectorizer.fit_transform(documents)
```

Here we've created count vector which contains the frequency of each word of each document. This is for finding the Cosine Similarity.

```
In [14]: #Creating a dataframe to store each count_vectorizer
doc_term_matrix = sparse_matrix.todense()
df = pd.DataFrame(doc_term_matrix,
                  columns=count_vectorizer.get_feature_names(),
                  index=['test1', 'test2', 'test3', 'test4', 'test5'])
df
```

```
Out[14]:
```

	accuracy	again	aimed	alokam	as	at	avoid	before	better	check	...	the	third	this	to	tokens	we	web	with	work	working
test1	1	0	0	0	0	0	0	0	0	1	...	1	0	1	1	0	0	0	0	0	0
test2	1	0	0	0	0	0	0	0	0	1	...	2	0	1	1	0	0	1	0	0	0
test3	1	0	0	1	0	0	0	0	0	1	...	1	1	1	1	0	0	0	0	0	0
test4	1	0	0	0	1	0	0	0	0	1	...	1	0	1	2	0	0	0	0	0	0
test5	0	1	1	0	0	1	1	1	1	0	...	3	0	2	2	1	1	0	1	1	1

5 rows × 50 columns

Here we've combined the count vectors of every document into Pandas Data Frame.

```
In [15]: #Printing the Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df, df))

[[1.          0.83770782 0.85485041 0.85202865 0.48374383]
 [0.83770782 1.          0.82353211 0.74586985 0.4982019 ]
 [0.85485041 0.82353211 1.          0.76477489 0.46217904]
 [0.85202865 0.74586985 0.76477489 1.          0.48368611]
 [0.48374383 0.4982019  0.46217904 0.48368611 1.          ]]
```

Here we printed Cosine Similarity of Every Document

Results and Output

Top words in Each Input Text file:

```
Top words in document 1
  Word: first, TF-IDF: 0.80472
  Word: Frequency, TF-IDF: 0.64378
  Word: accuracy, TF-IDF: 0.40236
Top words in document 2
  Word: Second, TF-IDF: 0.80472
  Word: accuracy.The, TF-IDF: 0.80472
  Word: course, TF-IDF: 0.80472
Top words in document 3
  Word: Third, TF-IDF: 0.80472
  Word: My, TF-IDF: 0.80472
  Word: Alokam, TF-IDF: 0.80472
Top words in document 4
  Word: fourth, TF-IDF: 0.80472
  Word: have, TF-IDF: 0.80472
  Word: as, TF-IDF: 0.80472
Top words in document 5
  Word: of, TF-IDF: 2.41416
  Word: number, TF-IDF: 1.60944
  Word: the, TF-IDF: 0.80472
```

Cosine similarity

```
[[1.          0.83770782 0.85485041 0.85202865 0.48374383]
 [0.83770782 1.          0.82353211 0.74586985 0.4982019 ]
 [0.85485041 0.82353211 1.          0.76477489 0.46217904]
 [0.85202865 0.74586985 0.76477489 1.          0.48368611]
 [0.48374383 0.4982019  0.46217904 0.48368611 1.          ]]
```