Harshit Mishra (19BCE0799)

CSE-3024 WEB MINING

LAB ASSIGNMENT 1

Question 1:

Aim: Tokenize an input text file without using NLTK and remove the stop words from the tokens.

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.
- Using split() function of NumPy lists we tokenize each token in our text and save it in a list.
- Then we use NumPy's unique method to only include unique tokens from our identified set of tokens.
- To remove stop words, we create a tentative list of stop words and using a nested for loop we check if the given token belongs to that list or not. If it doesn't then we save it else we discard it.
- Finally, we print our list that contains the resultant tokens post removal of stop words.

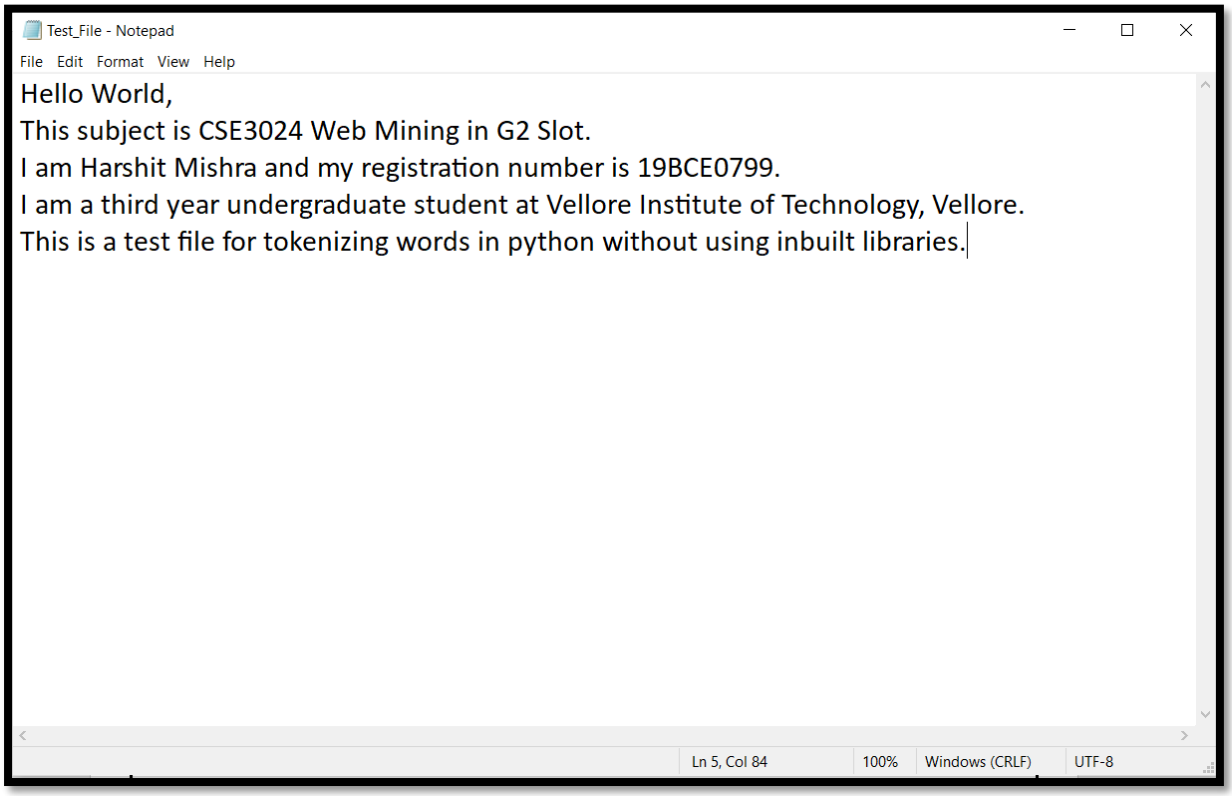
Harshit Mishra (19BCE0799)

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)
```

Harshit Mishra (19BCE0799)

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

Out[2]: ' Hello World This subject is CSE3024 Web Mining in G2 Slot I am Harshit Mishra
        and my registration number is 19BCE0799 I am a third year undergraduate student
        at Vellore Institute of Technology Vellore This is a test file for tokenizing w
        ords in python without using inbuilt libraries'
```

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

        ['Hello', 'World', 'This', 'subject', 'is', 'CSE3024', 'Web', 'Mining', 'in',
        'G2', 'Slot', 'I', 'am', 'Harshit', 'Mishra', 'and', 'my', 'registration', 'num
        ber', 'is', '19BCE0799', 'I', 'am', 'a', 'third', 'year', 'undergraduate', 'stu
        dent', 'at', 'Vellore', 'Institute', 'of', 'Technology', 'Vellore', 'This', 'i
        s', 'a', 'test', 'file', 'for', 'tokenizing', 'words', 'in', 'python', 'withou
        t', 'using', 'inbuilt', 'libraries']
```

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

        ['19BCE0799' 'CSE3024' 'G2' 'Harshit' 'Hello' 'I' 'Institute' 'Mining'
         'Mishra' 'Slot' 'Technology' 'This' 'Vellore' 'Web' 'World' 'a' 'am'
         'and' 'at' 'file' 'for' 'in' 'inbuilt' 'is' 'libraries' 'my' 'number'
         'of' 'python' 'registration' 'student' 'subject' 'test' 'third'
         'tokenizing' 'undergraduate' 'using' 'without' 'words' 'year']
```

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 [5]: #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 [6]: #Printing cleaned tokens in our input text file
print(res)
```

```
['19bce0799', 'cse3024', 'file', 'g2', 'harshit', 'hello', 'inbuilt', 'institut
e', 'libraries', 'mining', 'mishra', 'number', 'python', 'registration', 'slo
t', 'student', 'subject', 'technology', 'test', 'third', 'tokenizing', 'undergr
aduate', 'using', 'vellore', 'web', 'without', 'words', 'world', '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:

- Input text:

```
Hello World,
This subject is CSE3024 Web Mining in G2 Slot.
I am Harshit Mishra and my registration number is 19BCE0799.
I am a third year undergraduate student at Vellore Institute of Technology, Vellore.
This is a test file for tokenizing words in python without using inbuilt libraries.
```

- Tokens of input text:

```
['19bce0799' 'a' 'am' 'and' 'at' 'cse3024' 'file' 'for' 'g2' 'harshit'
 'hello' 'i' 'in' 'inbuilt' 'institute' 'is' 'libraries' 'mining' 'mishra'
 'my' 'number' 'of' 'python' 'registration' 'slot' 'student' 'subject'
 'technology' 'test' 'third' 'this' 'tokenizing' 'undergraduate' 'using'
 'vellore' 'web' 'without' 'words' 'world' 'year']
```

- Tokens after removal of stop words:

```
['19bce0799', 'cse3024', 'file', 'g2', 'harshit', 'hello', 'inbuilt', 'institut
e', 'libraries', 'mining', 'mishra', 'number', 'python', 'registration', 'slo
t', 'student', 'subject', 'technology', 'test', 'third', 'tokenizing', 'undergr
aduate', 'using', 'vellore', 'web', 'without', 'words', 'world', 'year']
```

- We can use same code for tokenization without NLTK to text input… Here we have used text file input but instead of reading that input and assigning it to a string variable, we can directly input it to string variable text and execute the exact same code to tokenize it.
- For 2a and 2b we are going to use same code. We will change the input files.

Harshit Mishra (19BCE0799)

Question 2

2a) Aim: Tokenize a single sentence text without using NLTK.

2b) Aim: Tokenize a multi-line text without using NLTK.

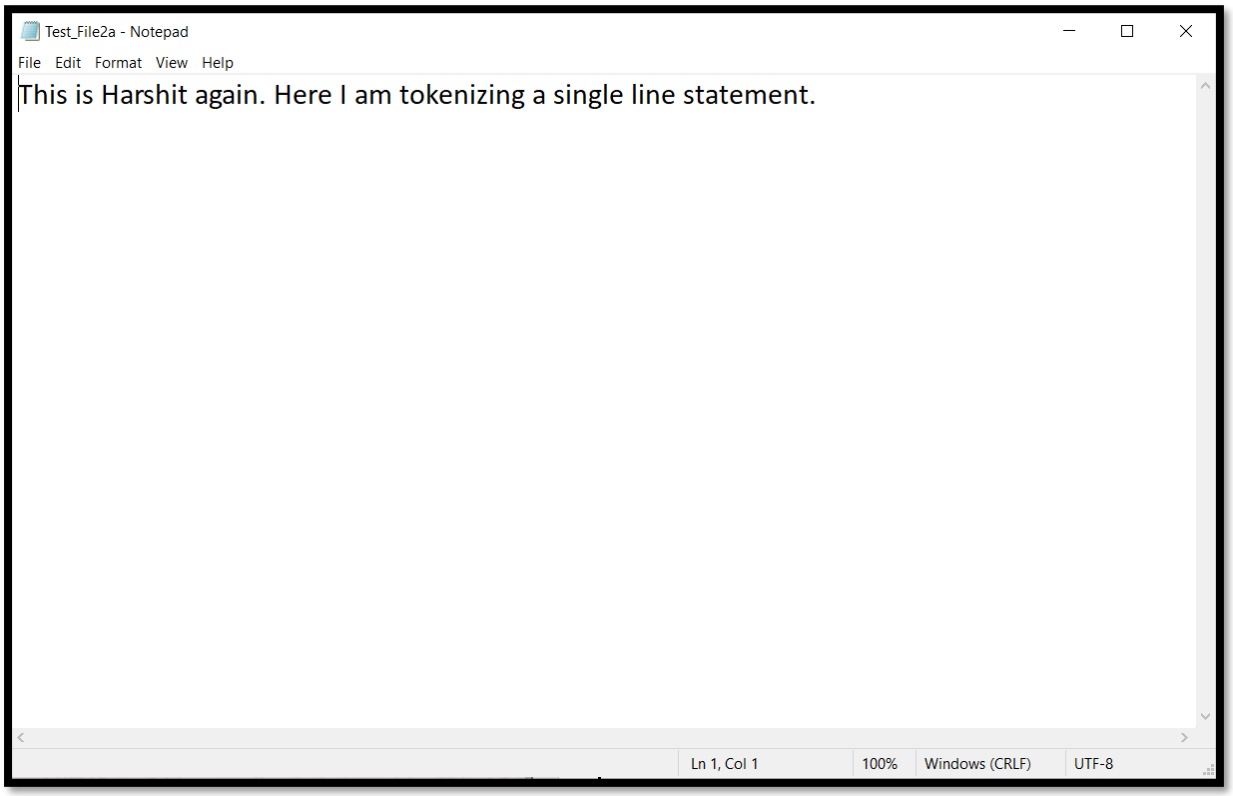
Procedure:

- Firstly, we import the text file into our workspace. To do this we can use open method of python which reads our text file to the workspace.
- Next, we read each word in the input file as a string input. This can be implemented using nested for loop wherein we split each word whenever we encounter a space.
- Next, using regex we remove the punctuations from our input string. This will render token a better syntactic structure and break the sentence bonds.
- Next, we split and store each token into a list using split method of python.
- Then finally we remove stop words as we did in previous assignment.

Code here is same as previous one… just that input file is different…

Harshit Mishra (19BCE0799)

2a) Input Text File:



Harshit Mishra (19BCE0799)

2a) 2Code Snippets and Output:

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.lower()
```

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

Out[2]:
```
' this is harshit again here i am tokenizing a single line statement'
```

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

```
['this', 'is', 'harshit', 'again', 'here', 'i', 'am', 'tokenizing', 'a', 'sin
gle', 'line', 'statement']
```

In [4]:
```
#Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)
```

```
['a' 'again' 'am' 'harshit' 'here' 'i' 'is' 'line' 'single' 'statement'
 'this' 'tokenizing']
```

In [5]:
```
#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 [6]:
```
#Printing cleaned tokens in our input text file
print(res)
```

```
['again', 'harshit', 'here', 'line', 'single', 'statement', 'tokenizing']
```

2a) Result and output:

1) Input Sentence:

```
This is Harshit again. Here I am tokenizing a single line statement.
```

2) Token without removing stop words:

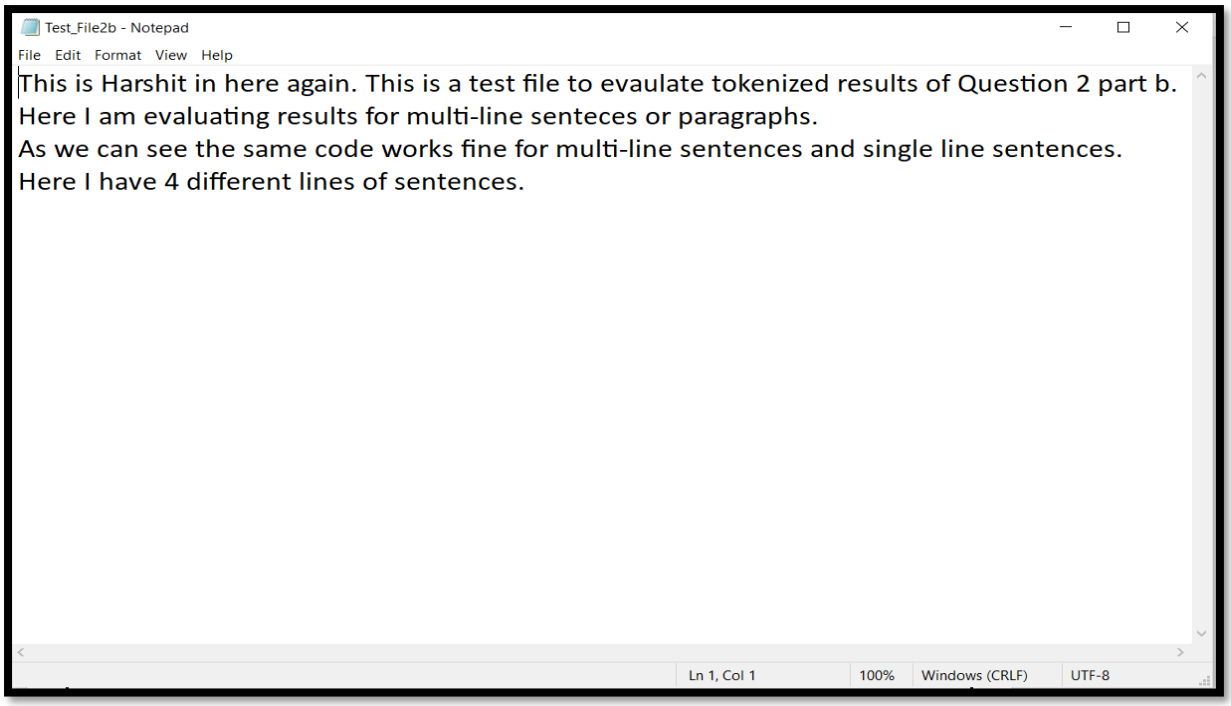
```
['a' 'again' 'am' 'harshit' 'here' 'i' 'is' 'line' 'single' 'statement'
 'this' 'tokenizing']
```

3) Token after removing stop words:

```
['again', 'harshit', 'here', 'line', 'single', 'statement', 'tokenizing']
```

2b) Input text file:

Harshit Mishra (19BCE0799)



2b) Code and Output:

Harshit Mishra (19BCE0799)

In [1]:
```
#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()
```

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

Out[2]:
```
' this is harshit in here again this is a test file to evaulate tokenized res
ults of question 2 part b here i am evaluating results for multiline senteces
or paragraphs as we can see the same code works fine for multiline sentences
and single line sentences here i have 4 different lines of sentences'
```

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

```
['this', 'is', 'harshit', 'in', 'here', 'again', 'this', 'is', 'a', 'test',
'file', 'to', 'evaulate', 'tokenized', 'results', 'of', 'question', '2', 'par
t', 'b', 'here', 'i', 'am', 'evaluating', 'results', 'for', 'multiline', 'sen
teces', 'or', 'paragraphs', 'as', 'we', 'can', 'see', 'the', 'same', 'code',
'works', 'fine', 'for', 'multiline', 'sentences', 'and', 'single', 'line', 's
entences', 'here', 'i', 'have', '4', 'different', 'lines', 'of', 'sentences']
```

In [4]:
```
#Printing unique tokens
import numpy as np
tokens = np.unique(tokens)
print(tokens)
```

```
['2' '4' 'a' 'again' 'am' 'and' 'as' 'b' 'can' 'code' 'different'
 'evaluating' 'evaulate' 'file' 'fine' 'for' 'harshit' 'have' 'here' 'i'
 'in' 'is' 'line' 'lines' 'multiline' 'of' 'or' 'paragraphs' 'part'
 'question' 'results' 'same' 'see' 'senteces' 'sentences' 'single' 'test'
 'the' 'this' 'to' 'tokenized' 'we' 'works']
```

In [5]:
```
#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 [6]:
```
#Printing cleaned tokens in our input text file
print(res)
```

```
['2', '4', 'again', 'as', 'b', 'can', 'code', 'different', 'evaluating', 'eva
ulate', 'file', 'fine', 'harshit', 'have', 'here', 'line', 'lines', 'multilin
e', 'or', 'paragraphs', 'part', 'question', 'results', 'same', 'see', 'sentec
es', 'sentences', 'single', 'test', 'the', 'to', 'tokenized', 'we', 'works']
```

2b) Result and Output:

Harshit Mishra (19BCE0799)

1) Input Text:

```
This is Harshit in here again. This is a test file to evaulate tokenized results of Question 2 part b.
Here I am evaluating results for multi-line senteces or paragraphs.
As we can see the same code works fine for multi-line sentences and single line sentences.
Here I have 4 different lines of sentences.
```

2) Tokens without removing stop words:

```
['2' '4' 'a' 'again' 'am' 'and' 'as' 'b' 'can' 'code' 'different'
 'evaluating' 'evaulate' 'file' 'fine' 'for' 'harshit' 'have' 'here' 'i'
 'in' 'is' 'line' 'lines' 'multiline' 'of' 'or' 'paragraphs' 'part'
 'question' 'results' 'same' 'see' 'senteces' 'sentences' 'single' 'test'
 'the' 'this' 'to' 'tokenized' 'we' 'works']
```

3) Tokens after removing stop words:

```
['2', '4', 'again', 'as', 'b', 'can', 'code', 'different', 'evaluating', 'eva
ulate', 'file', 'fine', 'harshit', 'have', 'here', 'line', 'lines', 'multilin
e', 'or', 'paragraphs', 'part', 'question', 'results', 'same', 'see', 'sentec
es', 'sentences', 'single', 'test', 'the', 'to', 'tokenized', 'we', 'works']
```