

CSE3024 - WEB MINING (L41 + L42)

NAME – ANMOL
REG. NO. - 19BCE0891

DIGITAL ASSIGNMENT – 4

1. Create a Python programme that uses TF-IDF to find the important words in the given corpus. Note: Collect strings from the following documents and create a corpus containing strings from documents d1, d2, and d3.

- d1: VIT Vellore University
- d2: VIT
- d3: Web

The screenshot shows a Jupyter Notebook titled "Assignment_4" with a last checkpoint from Saturday at 9:53 PM. The notebook is running on a local host at 8888. The code in the first cell is as follows:

```
In [1]: # importing modules
from sklearn.feature_extraction.text import TfidfVectorizer

# Defining docs
d0 = 'VIT Vellore University'
d1 = 'VIT'
d2 = 'web'

string = [d0, d1, d2]
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)

print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ': ', ele2)

print('\nword indexes:')
print(tfidf.vocabulary_)

print('\ntf-idf value:')
print(result)

print('\ntf-idf values in matrix form:')
print(result.toarray())
```

The output of the code is displayed below the cell:

```
idf values:
university : 1.6931471885599454
vellore : 1.6931471885599454
vit : 1.2876828724517808
web : 1.6931471885599454

Word indexes:
{'vit': 2, 'vellore': 1, 'university': 0, 'web': 3}

tf-idf value:
(0, 0)      0.6227660078332259
(0, 1)      0.6227660078332259
(0, 2)      0.4736296010332684
(1, 2)      1.0
(2, 3)      1.0

tf-idf values in matrix form:
[[0.62276601 0.62276601 0.4736296  0.        ]
 [0.         0.         1.         0.        ]
 [0.         0.         1.         1.        ]]
```

CODE -

```
# importing modules
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Defining docs
d0 = 'VIT Vellore University'
d1 = 'VIT'
d2 = 'Web'
```

```
string = [d0, d1, d2]
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)
```

```
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ': ', ele2)
```

CSE3024 - WEB MINING (L41 + L42)

```
print("\nWord indexes:")
print(tfidf.vocabulary_)
```

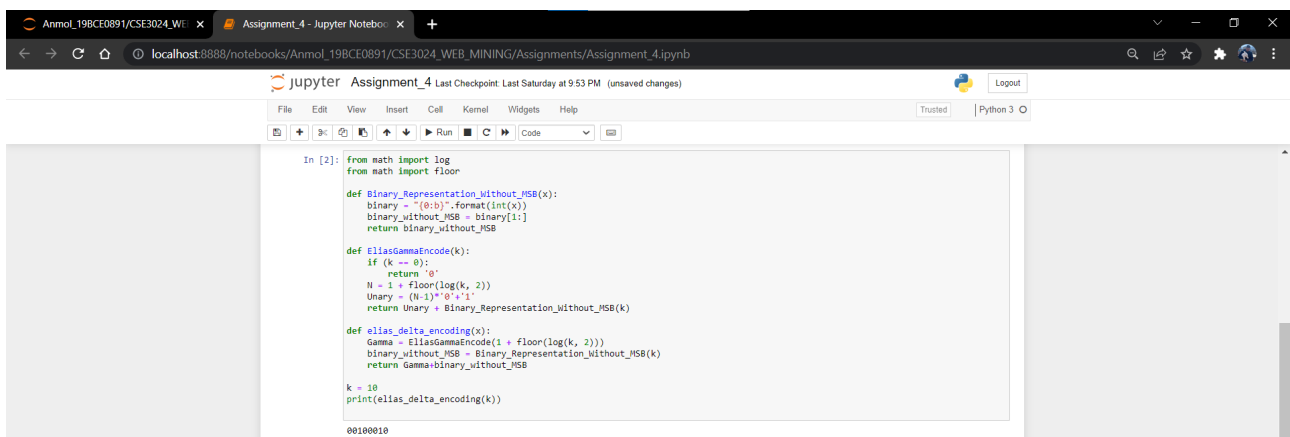
```
print("\ntf-idf value:")
print(result)
```

```
print("\ntf-idf values in matrix form:")
print(result.toarray())
```

2. Create a Python programme that performs Elias Delta Encoding and Decoding for a given number.

ENCODING :

METHOD - 1



CODE (method-1)

```
from math import log
from math import floor
```

```
def Binary_Representation_Without_MSB(x):
    binary = "{0:b}".format(int(x))
    binary_without_MSB = binary[1:]
    return binary_without_MSB
```

```
def EliasGammaEncode(k):
    if (k == 0):
        return '0'
    N = 1 + floor(log(k, 2))
    Unary = (N-1)*'0'+ '1'
    return Unary + Binary_Representation_Without_MSB(k)
```

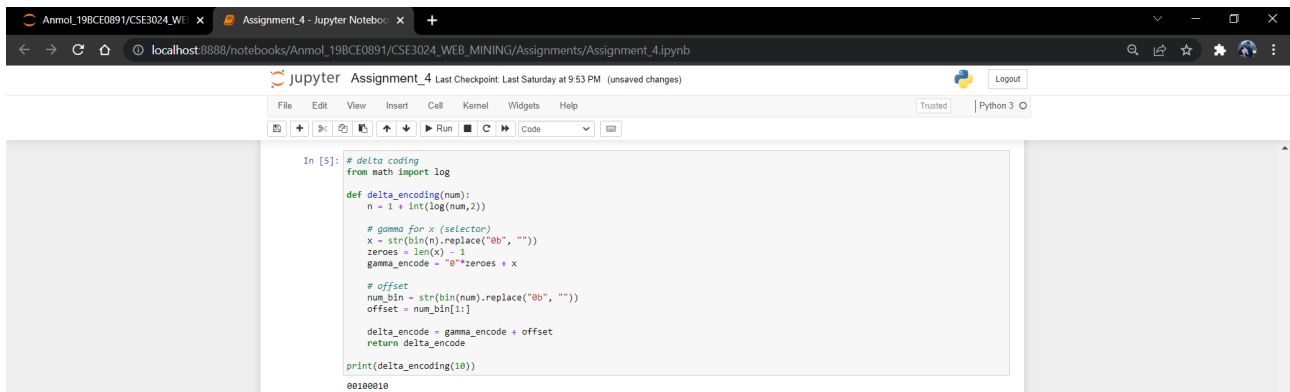
```
def elias_delta_encoding(x):
```

CSE3024 - WEB MINING (L41 + L42)

```
Gamma = EliasGammaEncode(1 + floor(log(k, 2)))  
binary_without_MSB = Binary_Representation_Without_MSB(k)  
return Gamma+binary_without_MSB
```

```
k = 10  
print(elias_delta_encoding(k))
```

METHOD – 2



The screenshot shows a Jupyter Notebook window titled 'Assignment_4'. The code cell contains the following Python code:

```
In [5]: # delta coding  
from math import log  
  
def delta_encoding(num):  
    n = 1 + int(log(num,2))  
  
    # gamma for x (selector)  
    x = str(bin(n).replace("0b", ""))  
    zeroes = len(x) - 1  
    gamma_encode = "0"*zeroes + x  
  
    # offset  
    num_bin = str(bin(num).replace("0b", ""))  
    offset = num_bin[1:]  
  
    delta_encode = gamma_encode + offset  
    return delta_encode  
  
print(delta_encoding(10))  
  
00100010
```

CODE (method -2)

```
# delta coding  
from math import log
```

```
def delta_encoding(num):  
    n = 1 + int(log(num,2))
```

```
    # gamma for x (selector)  
    x = str(bin(n).replace("0b", ""))  
    zeroes = len(x) - 1  
    gamma_encode = "0"*zeroes + x
```

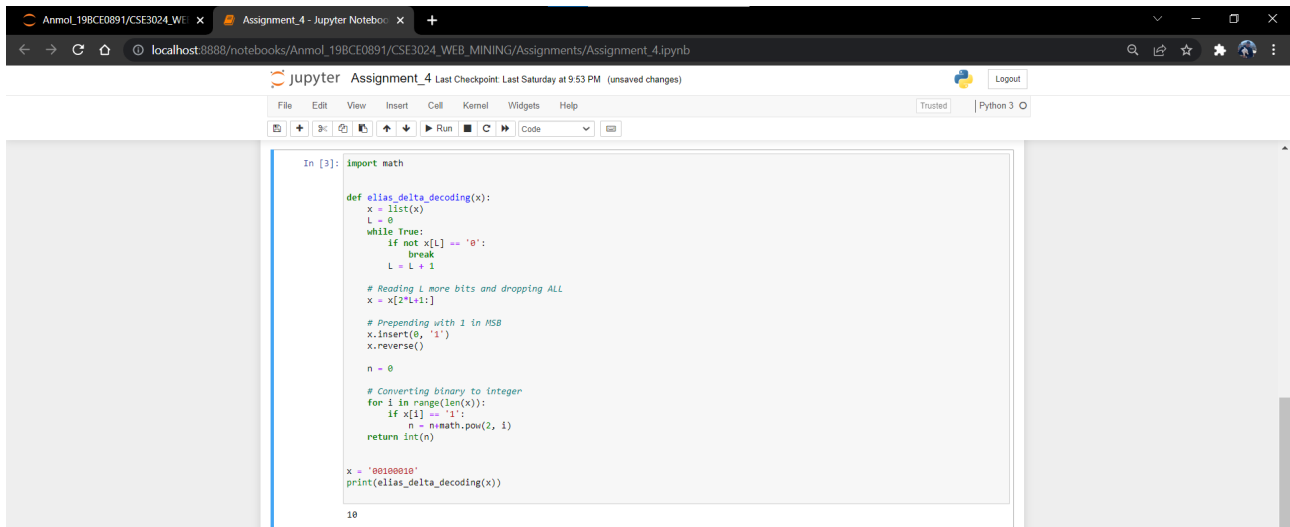
```
    # offset  
    num_bin = str(bin(num).replace("0b", ""))  
    offset = num_bin[1:]
```

```
    delta_encode = gamma_encode + offset  
    return delta_encode
```

```
print(delta_encoding(10))
```

CSE3024 - WEB MINING (L41 + L42)

DECODING

A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_4.ipynb'. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The main area displays a code cell with the following Python code:

```
In [3]: import math

def elias_delta_decoding(x):
    x = list(x)
    L = 0
    while True:
        if not x[L] == '0':
            break
        L = L + 1

    # Reading L more bits and dropping ALL
    x = x[2*L+1:]

    # Prepending with 1 in MSB
    x.insert(0, '1')
    x.reverse()

    n = 0

    # Converting binary to integer
    for i in range(len(x)):
        if x[i] == '1':
            n = n+math.pow(2, i)
    return int(n)

x = '00100010'
print(elias_delta_decoding(x))
```

The output of the code is '10'.

CODE -

```
import math
```

```
def elias_delta_decoding(x):
```

```
    x = list(x)
```

```
    L = 0
```

```
    while True:
```

```
        if not x[L] == '0':
```

```
            break
```

```
        L = L + 1
```

```
    # Reading L more bits and dropping ALL
```

```
    x = x[2*L+1:]
```

```
    # Prepending with 1 in MSB
```

```
    x.insert(0, '1')
```

```
    x.reverse()
```

```
    n = 0
```

```
    # Converting binary to integer
```

```
    for i in range(len(x)):
```

```
        if x[i] == '1':
```

```
            n = n+math.pow(2, i)
```

```
    return int(n)
```

```
x = '00100010'
```

```
print(elias_delta_decoding(x))
```