Terna Engineering College
Computer Engineering Department
Program: Sem VIII

Course: Natural Language Processing

Experiment No. 7

A.1 Aim: Implement the Viterbi algorithm using python or NLTK.

PART B
(PART B: TO BE COMPLETED BY STUDENTS)

Roll No. 50	**Name:** AMEY THAKUR
Class: BE COMPS B	**Batch:** B3
Date of Experiment: 01/04/2022	**Date of Submission:** 01/04/2022
Grade:	

B.1 Software Code written by a student:

```
import numpy as np
from numba import jit

@jit(nopython=True)
def viterbi(A, C, B, O):

    I = A.shape[0]
    N = len(O)

    D = np.zeros((I, N))
    E = np.zeros((I, N-1)).astype(np.int32)
    D[:, 0] = np.multiply(C, B[:, O[0]])

    for n in range(1, N):
        for i in range(I):
            temp_product = np.multiply(A[:, i], D[:, n-1])
            D[i, n] = np.max(temp_product) * B[i, O[n]]
            E[i, n-1] = np.argmax(temp_product)

    S_opt = np.zeros(N).astype(np.int32)
    S_opt[-1] = np.argmax(D[:, -1])
    for n in range(N-2, -1, -1):
        S_opt[n] = E[int(S_opt[n+1]), n]
```

```
    return S_opt, D, E

    A = np.array([[0.8, 0.1, 0.1], [0.2, 0.7, 0.1], [0.1, 0.3, 0.6]])
    C = np.array([0.6, 0.2, 0.2])
    B = np.array([[0.7, 0.0, 0.3], [0.1, 0.9, 0.0], [0.0, 0.2, 0.8]])

    O = np.array([0, 2, 0, 2, 2, 1]).astype(np.int32)
    S_opt, D, E = viterbi(A, C, B, O)
    print('Observation sequence: O = ', O)
    print('Optimal state sequence: S = ', S_opt)
    np.set_printoptions(formatter={'float': "{: 7.4f}".format})
    print('D =', D, sep='\n')
    np.set_printoptions(formatter={'float': "{: 7.0f}".format})
    print('E =', E, sep='\n')
```

B.2 Input and Output:

```
import numpy as np
from numba import jit

@jit(nopython=True)
def viterbi(A, C, B, O):

    I = A.shape[0]
    N = len(O)

    D = np.zeros((I, N))
    E = np.zeros((I, N-1)).astype(np.int32)
    D[:, 0] = np.multiply(C, B[:, O[0]])

    for n in range(1, N):
        for i in range(I):
            temp_product = np.multiply(A[:, i], D[:, n-1])
            D[i, n] = np.max(temp_product) * B[i, O[n]]
            E[i, n-1] = np.argmax(temp_product)

    S_opt = np.zeros(N).astype(np.int32)
    S_opt[-1] = np.argmax(D[:, -1])
    for n in range(N-2, -1, -1):
        S_opt[n] = E[int(S_opt[n+1]), n]

    return S_opt, D, E

A = np.array([[0.8, 0.1, 0.1], [0.2, 0.7, 0.1], [0.1, 0.3, 0.6]])
C = np.array([0.6, 0.2, 0.2])
B = np.array([[0.7, 0.0, 0.3], [0.1, 0.9, 0.0], [0.0, 0.2, 0.8]])

O = np.array([0, 2, 0, 2, 2, 1]).astype(np.int32)
S_opt, D, E = viterbi(A, C, B, O)
print('Observation sequence: O = ', O)
print('Optimal state sequence: S = ', S_opt)
np.set_printoptions(formatter={'float': "{: 7.4f}".format})
print('D =', D, sep='\n')
np.set_printoptions(formatter={'float': "{: 7.0f}".format})
print('E =', E, sep='\n')
```

```
Observation sequence: O =  [0 2 0 2 2 1]
Optimal state sequence: S =  [0 0 0 2 2 1]
D =
[[4.20000000e-01 1.00800000e-01 5.64480000e-02 1.35475200e-02
  3.25140480e-03 0.00000000e+00]
 [2.00000000e-02 0.00000000e+00 1.00800000e-03 0.00000000e+00
  0.00000000e+00 5.85252864e-04]
 [0.00000000e+00 3.36000000e-02 0.00000000e+00 4.51584000e-03
  2.16760320e-03 2.60112384e-04]]
E =
[[0 0 0 0 0]
 [0 0 0 0 2]
 [0 2 0 2 2]]
```

B.3 Observations and learning:

- The Viterbi algorithm is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).
- The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, speech synthesis, diarization, keyword spotting, computational linguistics, and bioinformatics.
- For example, in speech-to-text (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal.

B.4 Conclusion:

We have successfully Implemented the Viterbi algorithm using python or NLTK.