Terna Engineering College
Computer Engineering Department
Program: Sem VIII

Course: Natural Language Processing

Experiment No. 5

A.1 Aim: Implement a bi-gram model for 3 sentences 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: 21/02/2022	**Date of Submission:** 21/02/2022
Grade:	

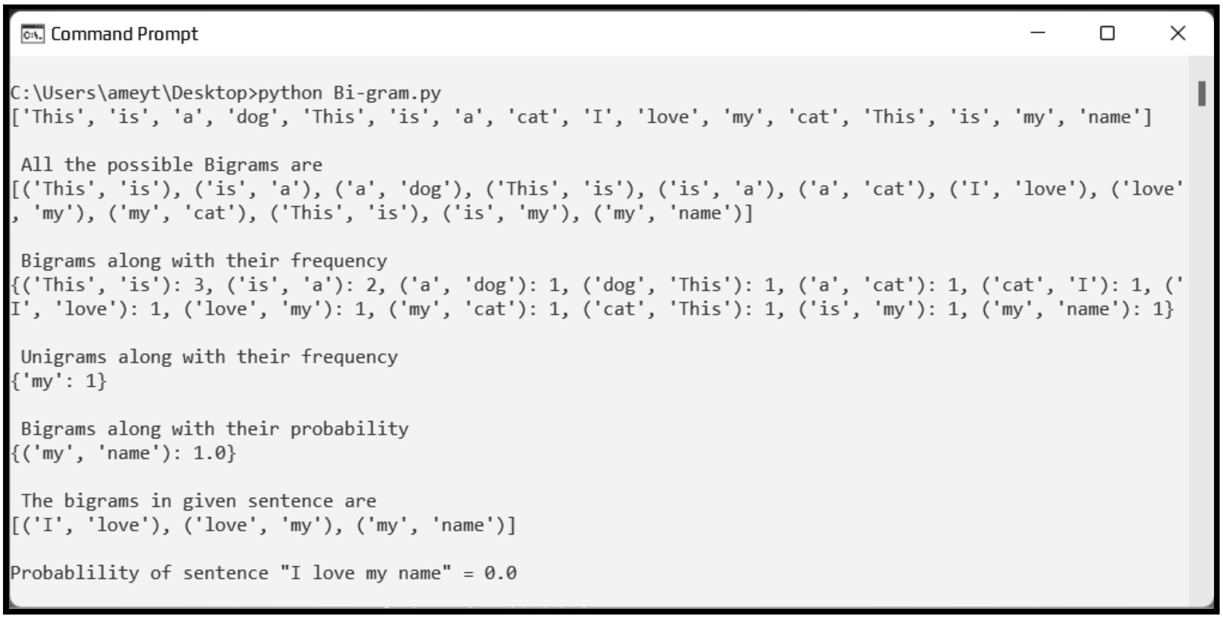
B.1 Software Code written by a student:

- **Bi-gram.py**

```
def readData():
    data = ['This is a  dog','This is a cat','I love my cat','This is my name ']
    dat=[]
    for i in range(len(data)):
        for word in data[i].split():
            dat.append(word)
    print(dat)
    return dat

def createBigram(data):
    listOfBigrams = []
    bigramCounts = {}
    unigramCounts = {}
    for i in range(len(data)-1):
        if i<len(data) - 1 and data[i+1].islower():
            listOfBigrams.append((data[i], data[i + 1]))
        if (data[i], data[i+1]) in bigramCounts:
            bigramCounts[(data[i], data[i + 1])] += 1
        else:
            bigramCounts[(data[i], data[i + 1])] = 1
    if data[i] in unigramCounts:
        unigramCounts[data[i]] += 1
    else:
        unigramCounts[data[i]] = 1
    return listOfBigrams, unigramCounts, bigramCounts


def calcBigramProb(listOfBigrams, unigramCounts, bigramCounts):
    listOfProb = {}
    for bigram in listOfBigrams:
        word1 = bigram[0]
        word2 = bigram[1]
    listOfProb[bigram] = (bigramCounts.get(bigram))/(unigramCounts.get(word1))
    return listOfProb


if __name__ == '__main__':
    data = readData()
listOfBigrams, unigramCounts, bigramCounts = createBigram(data)

print("\n All the possible Bigrams are ")
print(listOfBigrams)

print("\n Bigrams along with their frequency ")
print(bigramCounts)

print("\n Unigrams along with their frequency ")
print(unigramCounts)

bigramProb = calcBigramProb(listOfBigrams, unigramCounts, bigramCounts)

print("\n Bigrams along with their probability ")
print(bigramProb)
inputList="I love my name"
splt=inputList.split()
outputProb1 = 1
bilist=[]
bigrm=[]

for i in range(len(splt) - 1):
    if i<len(splt) - 1:
        bilist.append((splt[i], splt[i + 1]))
```

```
print("\n The bigrams in given sentence are ")
print(bilist)
for i in range(len(bilist)):
   if bilist[i] in bigramProb:
      outputProb1 *= bigramProb[bilist[i]]
   else:
      outputProb1 *= 0
print('\n' + 'Probablility of sentence \"I love my name\" = ' + str(outputProb1))
```

B.2 Input and Output:



```
C:\Users\ameyt\Desktop>python Bi-gram.py
['This', 'is', 'a', 'dog', 'This', 'is', 'a', 'cat', 'I', 'love', 'my', 'cat', 'This', 'is', 'my', 'name']

 All the possible Bigrams are
[('This', 'is'), ('is', 'a'), ('a', 'dog'), ('This', 'is'), ('is', 'a'), ('a', 'cat'), ('I', 'love'), ('love'
, 'my'), ('my', 'cat'), ('This', 'is'), ('is', 'my'), ('my', 'name')]

 Bigrams along with their frequency
{('This', 'is'): 3, ('is', 'a'): 2, ('a', 'dog'): 1, ('dog', 'This'): 1, ('a', 'cat'): 1, ('cat', 'I'): 1, ('
I', 'love'): 1, ('love', 'my'): 1, ('my', 'cat'): 1, ('cat', 'This'): 1, ('is', 'my'): 1, ('my', 'name'): 1}

 Unigrams along with their frequency
{'my': 1}

 Bigrams along with their probability
{('my', 'name'): 1.0}

 The bigrams in given sentence are
[('I', 'love'), ('love', 'my'), ('my', 'name')]

Probablility of sentence "I love my name" = 0.0
```

B.3 Observations and learning:

When you use a bigram model to predict the conditional probability of the next word, you are thus making the following approximation:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

B.4 Conclusion:

Using python or NLTK, we were successful in implementing a bi-gram model for three texts.