Name : Yash Sarang	Class/Roll No. : D16AD / 47	Grade :

Title of Experiment :
Understanding N-grams in Natural Language Processing (NLP)

Problem Statement :
The problem is to investigate and understand the concept of n-grams in natural language processing (NLP) and how they can be effectively used in language modeling, text analysis, and predictive tasks. The objective is to explore the benefits and applications of n-grams in NLP.

Description / Theory :
N-grams are contiguous sequences of n items (words, characters, or symbols) from a given text. In NLP, n-grams are widely used to model language at the syntactic and semantic levels. Unigrams (n=1) represent individual words, bigrams (n=2) are pairs of adjacent words, trigrams (n=3) consist of three adjacent words, and so on. N-grams play a crucial role in various NLP tasks such as machine translation, speech recognition, sentiment analysis, and more.

Flowchart :
1. Input a text corpus.
2. Preprocess the text (tokenization, lowercasing, etc.).
3. Generate n-grams of varying n (unigrams, bigrams, trigrams, etc.).
4. Analyze the n-grams for frequency and distribution.
5. Use n-grams for language modeling or other NLP tasks.
6. Evaluate the effectiveness of n-grams in the specific task.

Program:

```
import nltk
nltk.download('punkt')
d=input("Enter corpus = ")
```

```
Enter corpus = Buba is good to play and watch and cum on
```

```
def preprocess(d):
    d=d.lower()
    d="eos "+ d
    d=d.replace("."," eos")
    return d
d=preprocess(d)
print("Preprocessed Data corpus = \n",d)
```

```
Preprocessed Data corpus =
eos buba is good to play and watch and cum on
```

```
from nltk import word_tokenize
def generate_tokens(d):
    tokens = word_tokenize(d)
    return tokens
tokens = generate_tokens(d)
distinct_tokens = list(set(sorted(tokens)))
print("Tokens in the corpus = \n",distinct_tokens)
```

```
Tokens in the corpus = ['play', 'and', 'cum', 'to', 'good', 'eos', 'watch', 'on',
'is', 'buba']
```

```
def generate_tokens_freq(tokens):
    dct={}
    for i in tokens:
        dct[i]=0
    for i in tokens:
        dct[i]+=1
    return dct
dct=generate_tokens_freq(tokens)
print("Frequency of each tokens = ")
for i in dct.items():
    print(i[0],"\t:" , i[1])
```

```
Frequency of each tokens =
eos     : 1
buba    : 1
is      : 1
good    : 1
to      : 1
play    : 1
and     : 2
watch   : 1
cum     : 1
on      : 1
```

```
def generate_ngrams(tokens,k):
    l=[]
    i=0
    while(i<len(tokens)):
        l.append(tokens[i:i+k])
        i=i+1
    l=l[:-1]
    return l
bigram = generate_ngrams(tokens,2)
print("N-grams generated (Here n is 2) = ")
for i in bigram:
    print(i)
```

```
N-grams generated (Here n is 2) =
['eos', 'buba']
['buba', 'is']
['is', 'good']
['good', 'to']
['to', 'play']
['play', 'and']
['and', 'watch']
['watch', 'and']
['and', 'cum']
['cum', 'on']
```

```
def generate_ngram_freq(bigram):
    dct1={}
    for i in bigram:
        st=" ".join(i)
        dct1[st]=0
    for i in bigram:
        st=" ".join(i)
        dct1[st]+=1
    return dct1
```

```
dct1=generate_ngram_freq(bigram)
print("Frequency of n-grams = ")
for i in dct1.items():
    print(i[0], ":", i[1])
```

```
Frequency of n-grams =
eos buba : 1
buba is : 1
is good : 1
good to : 1
to play : 1
play and : 1
and watch : 1
watch and : 1
and cum : 1
cum on : 1
```

```
def find1(s,dct1):
    try:
        return dct1[s]
    except:
        return 0
def print_probability_table(distinct_tokens,dct,dct1):
    n=len(distinct_tokens)
    l=[[]*n for i in range(n)]
    for i in range(n):
        denominator = dct[distinct_tokens[i]]
        for j in range(n):
            numerator = find1(distinct_tokens[i]+" "+distinct_tokens[j],dct1)
            l[i].append(float("{:.3f}".format(numerator/denominator)))
    return l
print("Probability table = \n")
probability_table=print_probability_table(distinct_tokens,dct,dct1)
n=len(distinct_tokens)
print("\t", end="")
for i in range(n):
    print(distinct_tokens[i],end="\t")
print("\n")
for i in range(n):
    print(distinct_tokens[i],end="\t")
    for j in range(n):
        print(probability_table[i][j],end="\t")
    print("\n")
```

Output:

```
Probability table =
      play   and   cum   to    good  eos   watch on    is    buba
play  0.0    1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
and   0.0    0.0   0.5   0.0   0.0   0.0   0.5   0.0   0.0   0.0
cum   0.0    0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0
 to   1.0    0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
good  0.0    0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
eos   0.0    0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0
watch 0.0    1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
on    0.0    0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
is    0.0    0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0
buba  0.0    0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0
```

Results and Discussions :

The results will demonstrate the n-grams generated for the given text, their frequency, and distribution. The discussion will revolve around the impact of n-grams on language modeling, their role in capturing context and relationships between words, and their usefulness in predictive modeling.

Conclusion:

N-grams, a fundamental concept in NLP, provide valuable insights into language structure and help in various language-related tasks. They enable us to analyze and model text effectively, capturing relationships between words and context. Understanding n-grams is essential for building more accurate and context-aware NLP models and applications.

******