| **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])**    **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)**    **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])**    **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.

**\*\*\*\*\*\***