|  |  |
| --- | --- |
| **Department of Computer Engineering Class: B.E. (Computer) (Div- B) (Sem-VIII)** | |
| **Subject: Computational Lab-II (NLP)(CSL804)** | |
| **Sr. No.** | **Title of Experiment** |
| **4.** | **Design N-gram model for any sample text.** |

|  |
| --- |
| **Course Name:** Computational Lab-II(NLP) |
| **Course Code:**  CSL804 |
| **Experiment No.:** 04 |
| **Lab outcome:** Acquire practical knowledge within the chosen area of technology for project development. |
| **Name of Student:** Moin Memon |
| **Student Roll No.:**275 |
| **Year/Semester/Div:** B.E./VIII/B |

**Experiment No. 04**

**Aim:** Design N-gram model for any sample text.

**Theory:**

**Language modelling:**

Language modeling is the way of determining the probability of any sequence of words. Language modeling is used in a wide variety of applications such as Speech Recognition, Spam filtering, etc. In fact, language modeling is the key aim behind the implementation of many state-of-the-art Natural Language Processing models.

**Types of Language modelling:**

* **Statistical Language Modeling’s:** Statistical Language Modeling, or Language Modeling, is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede. **Example:** N-gram language modeling.

* **Neural Language Modeling’s:** Neural network methods are achieving better results than classical methods both on standalone language models and when models are incorporated into larger models on challenging tasks like speech recognition and machine translation. A way of performing a neural language model is through word embeddings.

**N-gram Language model:**

N-gram can be defined as the contiguous sequence of n items from a given sample of text or speech. The items can be letters, words, or base pairs according to the application. It’s can also say to be as probabilistic model that's trained on a corpus of text. Such models are useful in many NLP applications including speech recognition, machine translation and predictive text input. An N-gram model is built by counting how often word sequences occur in corpus text and then estimating the probabilities. Thus, an N-gram language model predicts the probability of a given N-gram within any sequence of words in the language.

Thus, N-grams can said to be as a sequence of N words. Now, let us take a look at the following examples.

1. San Francisco (is a 2-gram)
2. The Three Musketeers (is a 3-gram)
3. She stood up slowly (is a 4-gram)

A good N-gram model can predict the next word in the sentence i.e the value of p (w|h).

**How to calculate the value of p (w|h)?**

**Example:**

**Text:** All your dreams can come true if we have the courage to pursue them.

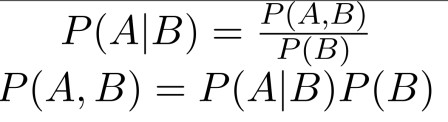
**Finding the probability of**  **P(** them **|** pursue them**) :**

**Generalizing the giving word into an equation we get:-**

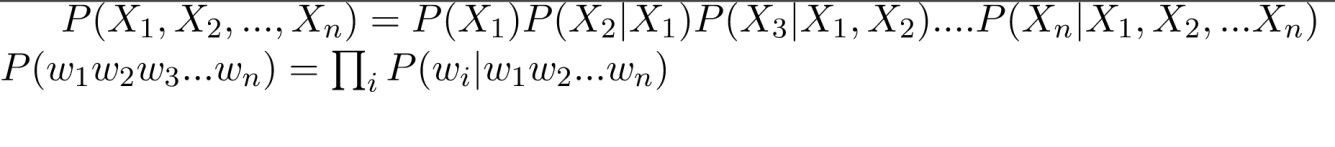
**P(** w3 **|** w1 w2 **) or P(**W**)**

**= P(** wn **|** w1 w2 w3 w4 ….wn )

Applying Chain rule of Probability, we get:



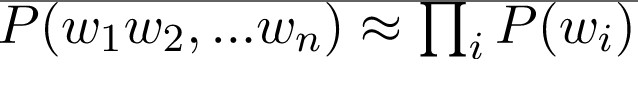
Simplifying given formula into Markov assumption, we get:



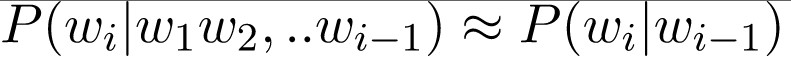
Therefore, the estimation of given probability function is written as, **For N-gram:**



**For unigram:**



**For bigram:**



**Consider wp and wn in the following text, wp = them**

**wn = pursue them**

For an Given text, **probability** (the number of times the previous word “wp” occurs before the word ‘wn’) / (the total number of times the previous word “wp” occurs in the corpus) = (Count (wp wn))/(Count (wp)).

**Now calculating the conditional probability, we get,**

**= (No. of times “pursue them” occurs ) / (No. of times “ them” occurs)**

**=1 /1**

**=1**

**Henceforth** we can say that whenever “Them” occurs, it will be followed by “Pursue” (This is because we have trained on a set of text and “Pursue” occurred only once in the context of “Pursue them”).

**Probability Estimation:**

Now, that we have understand the underlying base for N-gram models, we would think, how can we estimate the probability function. One of the most straightforward and intuitive ways to do so is Maximum Likelihood Estimation (MLE).

For example, to compute a particular bigram probability of a word y given a previous word x, you can determine the count of the bigram C(xy) and normalize it by the sum of all the bigrams that share the same first-word x.

**Sensitivity to the training corpus:**

* The N-gram model, like many statistical models, is significantly dependent on the training corpus. As a result, the probabilities often encode particular facts about a given training corpus. Besides, the performance of the Ngram model varies with the change in the value of N.

* Any N-gram that appeared a sufficient number of times might have a reasonable estimate for its probability. But because any corpus is limited, some perfectly acceptable English word sequences are bound to be missing from

it. As a result of it, the N-gram matrix for any training corpus is bound to have a substantial number of cases of putative “zero probability N-grams” **Program:**

import nltk

from nltk.util import ngrams

data ="All your dreams can come true if we have the courage to pursue them." tokens= nltk.word\_tokenize(data)

tokens

tokenized\_bigrams = list(nltk.bigrams(tokens))

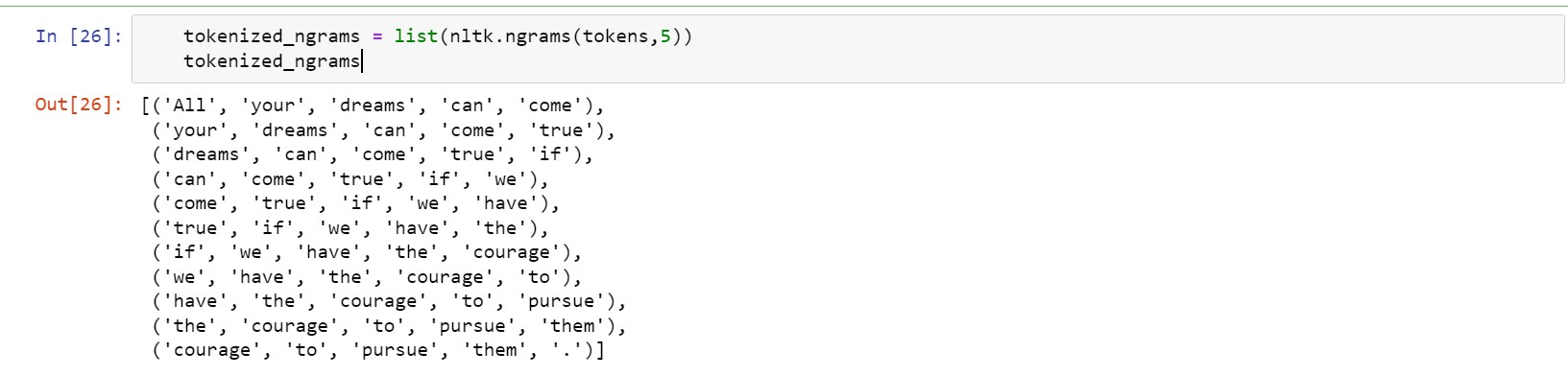
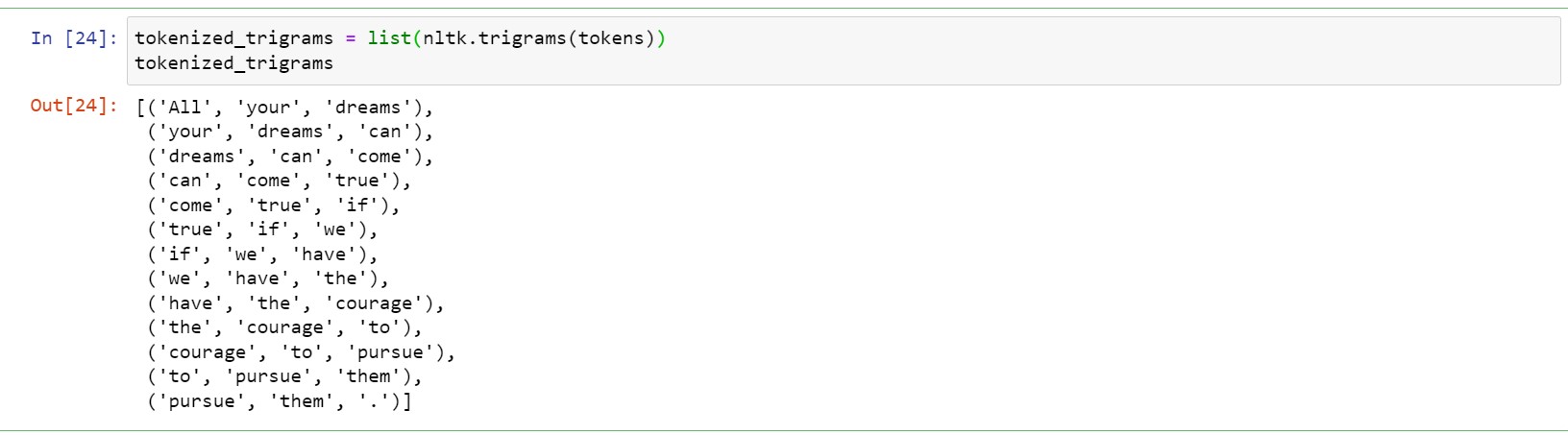
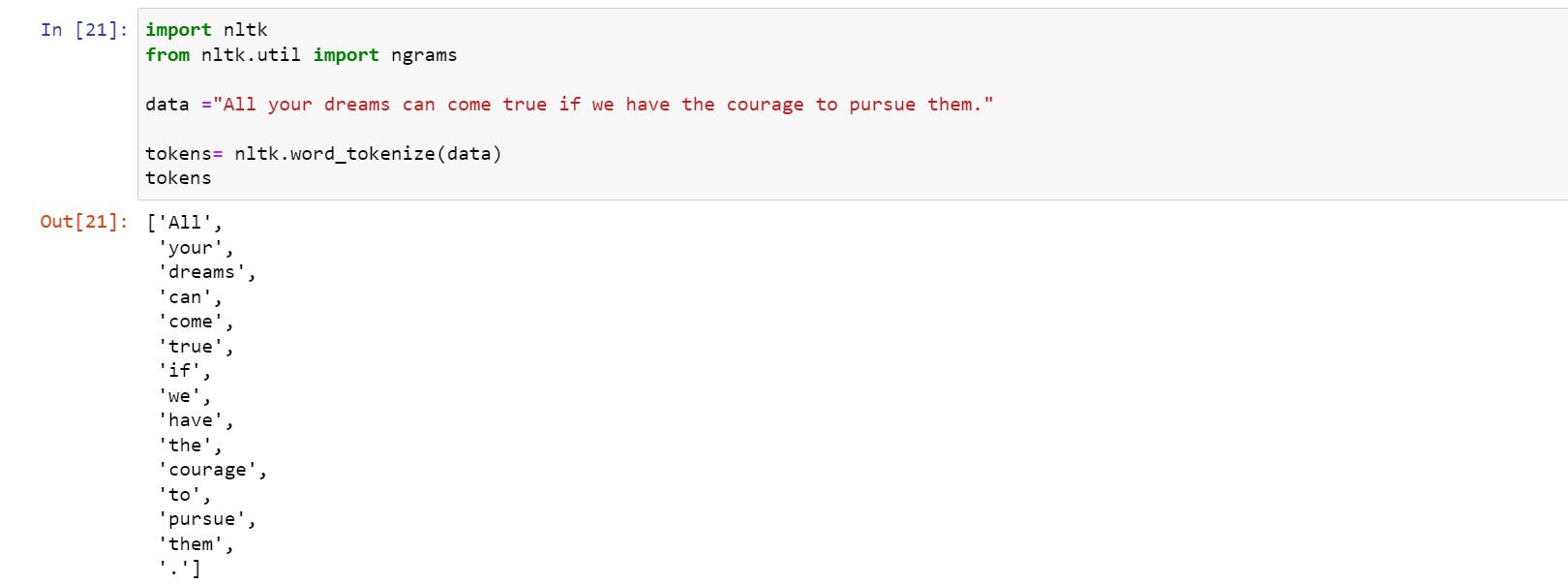
tokenized\_bigrams

tokenized\_trigrams = list(nltk.trigrams(tokens)) tokenized\_trigrams

tokenized\_ngrams = list(nltk.ngrams(tokens,5))

tokenized\_ngrams

**Output:**



**Conclusion:** Thus,We have successfully Demonstrated a N-gram model for any sample text.