Experiment – 5 (Building Bag of Words Model using NLTK)

Aim: To Implement bag of words model using NLTK

Algorithm:

1. **Import Libraries:**
   * Import necessary NLTK modules for tokenization, stop words, and frequency distribution.
2. **Preprocessing Function (preprocess\_text):**
   * Tokenizes, converts to lowercase, and removes stopwords from the input text.
3. **BoW Model Function (create\_bow\_model):**
   * Utilizes preprocess\_text for each text in the corpus.
   * Counts word frequencies using FreqDist and builds a Bag of Words model.
4. **Example Usage:**
   * An example list of texts is provided in the texts variable.
   * The create\_bow\_model function is called, and the resulting model is printed.

Code :

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.probability import FreqDist

nltk.download('punkt')

nltk.download('stopwords')

def preprocess\_text(text):

    stop\_words = set(stopwords.words('english'))

    word\_tokens = word\_tokenize(text)

    filtered\_words = [word.lower() for word in word\_tokens if word.isalpha() and word.lower() not in stop\_words]

    return filtered\_words

def create\_bow\_model(texts):

    all\_words = []

    for text in texts:

        words = preprocess\_text(text)

        all\_words.extend(words)

    word\_freq = FreqDist(all\_words)

    bow\_model = {word: freq for word, freq in word\_freq.items()}

    return bow\_model

# Example usage

texts = [

    "The cat sat on the mat, and the mat was comfortable.",

    "She sang a sweet song, a song that touched everyone's heart.",

    "Coding coding can be challenging, but coding is also incredibly rewarding.",

]

bow\_model = create\_bow\_model(texts)

# Print the Bag of Words model

print("Bag of Words Model:")

for word, freq in bow\_model.items():

    print(f"{word}: {freq}")

Output:

Bag of Words Model:

cat: 1

sat: 1

mat: 2

comfortable: 1

sang: 1

sweet: 1

song: 2

touched: 1

everyone: 1

heart: 1

coding: 3

challenging: 1

also: 1

incredibly: 1

rewarding: 1