Experiment-13

13) Write a NLP program to demonstrat following tasks

a)Tokenization, Removal of Stopword, Removal of punctuations,POS and NER Tagging

b)to demonstrate Bag of words,TF & IDF Vectrization, N-grams

a)Tokenization, Removal of Stopword, Removal of punctuations,POS and NER Tagging

In [ ]:

In [32]:
```
import nltk, spacy
from nltk.corpus import stopwords
from nltk import word_tokenize, pos_tag, ngrams
```

In [33]:
```
# Load spaCy model
nlp = spacy.load("en_core_web_sm")
```

In [34]:
```
text = "Apple is looking at buying U.K. startup for $1 billion in 2025."
```

In [35]:
```
#Tokenization → Breaking text into words
# Tokenization
tokens = word_tokenize(text)
print("Tokens:", tokens)
```

```
Tokens: ['Apple', 'is', 'looking', 'at', 'buying', 'U.K.', 'startup', 'for', '$',
'1', 'billion', 'in', '2025', '.']
```

In [36]:
```
#Stopword Removal → Filtering out common words
# Stopword Removal
filtered = [w for w in tokens if w.isalpha() and w.lower() not in stopwords.word
print("After Stopword Removal:", filtered)
```

```
After Stopword Removal: ['Apple', 'looking', 'buying', 'startup', 'billion']
```

In [37]:
```
import string

# Remove punctuation (just in case any remain)
tokens_no_punct = [w for w in filtered if w not in string.punctuation]

print("After Punctuation Removal:", tokens_no_punct)
```

```
After Punctuation Removal: ['Apple', 'looking', 'buying', 'startup', 'billion']
```

In [12]:
```
#POS Tagging → Identifying word roles (noun, verb, etc.)
# POS Tagging(Part of Speech)
```

```
print("POS Tags:", pos_tag(filtered))
```

```
POS Tags: [('Apple', 'NNP'), ('looking', 'VBG'), ('buying', 'VBG'), ('startup',
'NN'), ('billion', 'CD')]
```

In [13]:
```
#NER → Extracting entities like names, dates, money
# Named Entity Recognition
doc = nlp(text)
print("NER:", [(ent.text, ent.label_) for ent in doc.ents])
```

```
NER: [('Apple', 'ORG'), ('U.K.', 'GPE'), ('$1 billion', 'MONEY'), ('2025', 'DAT
E')]
```

b)to demonstrate Bag of words,TF & IDF Vectrization, N-grams

In [38]:
```
# ---- Imports ----
import nltk
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from nltk.util import ngrams
```

In [39]:
```
# Download tokenizer
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[39]: True

In [40]:
```
# Sample text
text = "Apple is looking at buying U.K. startup for $1 billion in 2025."
```

In [41]:
```
# ---- 1. Tokenization ----
tokens = word_tokenize(text)
print("Tokens:", tokens)
```

```
Tokens: ['Apple', 'is', 'looking', 'at', 'buying', 'U.K.', 'startup', 'for', '$',
'1', 'billion', 'in', '2025', '.']
```

In [42]:
```
# ---- 2. Bag of Words (BoW) ----
vectorizer = CountVectorizer()
bow = vectorizer.fit_transform([text])
print("\nBoW Words:", vectorizer.get_feature_names_out())
print("BoW Counts:", bow.toarray())
```

```
BoW Words: ['2025' 'apple' 'at' 'billion' 'buying' 'for' 'in' 'is' 'looking'
 'startup']
BoW Counts: [[1 1 1 1 1 1 1 1 1 1]]
```

In [43]:
```
# ---- 3. TF-IDF ----
tfidf_vectorizer = TfidfVectorizer()
tfidf = tfidf_vectorizer.fit_transform([text])
print("\nTF-IDF Words:", tfidf_vectorizer.get_feature_names_out())
print("TF-IDF Values:", tfidf.toarray())
```

```
TF-IDF Words: ['2025' 'apple' 'at' 'billion' 'buying' 'for' 'in' 'is' 'looking'
 'startup']
TF-IDF Values: [[0.31622777 0.31622777 0.31622777 0.31622777 0.31622777 0.3162277
7
  0.31622777 0.31622777 0.31622777 0.31622777]]
```

In [44]:

```
#N-grams → Generating word pairs/triples
# N-grams
print("Bigrams:", list(ngrams(filtered, 2)))
print("Trigrams:", list(ngrams(filtered, 3)))
```

```
Bigrams: [('Apple', 'looking'), ('looking', 'buying'), ('buying', 'startup'), ('s
tartup', 'billion')]
Trigrams: [('Apple', 'looking', 'buying'), ('looking', 'buying', 'startup'), ('bu
ying', 'startup', 'billion')]
```