**WEEK-2**

**AIM:**

Write a program that extracts the words (features) used in a sentence

**Description:**

1.Dealing with structured data attributes like numeric or categorical variables are usually

not as challenging as unstructured attributes like text and images.

2. In case of unstructured data like text documents, the first

challenge is dealing with the unpredictable nature of the syntax, format, and content of

the documents,which make it a challenge to extract useful information for building

models.

3.The second challenge is transforming these textual representations into numeric

representations that can be understood by MachineLearning algorithms.

there are two aspects to execute feature engineering on text data.

• Pre-processing and normalizing text

• Feature extraction and engineering

Bag of Words Model

This is perhaps one of the simplest yet effective schemes of vectorizing features from

unstructured text. The core principle of this model is to convert text documents into

numeric vectors. The dimension or size of each vector is N where N indicates all

possible distinct words across the corpus of documents.

**Code:**

import pandas as pd

import numpy as np

import re

import nltk

corpus = ['The sky is blue and beautiful.',

'Love this blue and beautiful sky!',

'The quick brown fox jumps over the lazy dog.',

'The brown fox is quick and the blue dog is lazy!',

'The sky is very blue and the sky is very beautiful today',

'The dog is lazy but the brown fox is quick!'

]

labels = ['weather', 'weather', 'animals', 'animals', 'weather', 'animals']

corpus = np.array(corpus)

corpus\_df = pd.DataFrame({'Document': corpus,

'Category': labels})

corpus\_df = corpus\_df[['Document', 'Category']]

corpus\_df

**Output:**

Document Category

0 The sky is blue and beautiful. weather

1 Love this blue and beautiful sky! weather

2 The quick brown fox jumps over the lazy dog. animals

3 The brown fox is quick and the blue dog is lazy! animals

4 The sky is very blue and the sky is very beaut... weather

5 The dog is lazy but the brown fox is quick! animals

CodeText

**Normalization**

**Code:**

nltk.download('stopwords')

**Output:**

[nltk\_data] Downloading package stopwords to /root/nltk\_data...

[nltk\_data] Unzipping corpora/stopwords.zip.

True

**Code:**

from nltk.corpus import stopwords

wpt = nltk.WordPunctTokenizer()

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

def normalize\_document(doc):

  doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc, re.I)

  doc = doc.lower()

  doc = doc.strip()

  tokens = wpt.tokenize(doc)

  filtered\_tokens = [token for token in tokens if token not in stop\_words]

  doc = ' '.join(filtered\_tokens)

  return doc

normalize\_corpus = np.vectorize(normalize\_document)

norm\_corpus = normalize\_corpus(corpus)

norm\_corpus

**Output:**

array(['sky blue beautiful', 'love blue beautiful sky', 'quick brown fox jumps lazy dog', 'brown fox quick blue dog lazy', 'sky blue sky beautiful today', 'dog lazy brown fox quick'], dtype='<U30')

**Bag of words model**

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(min\_df=0., max\_df=1.)

cv\_matrix = cv.fit\_transform(norm\_corpus)

cv\_matrix = cv\_matrix.toarray()

cv\_matrix

vocab = cv.get\_feature\_names()

pd.DataFrame(cv\_matrix, columns=vocab)

**Output:**

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get\_feature\_names is deprecated; get\_feature\_names is deprecated in 1.0 and will be removed in 1.2. Please use get\_feature\_names\_out instead.

warnings.warn(msg, category=FutureWarning)

|  | **beautiful** | **blue** | **brown** | **dog** | **fox** | **jumps** | **lazy** | **love** | **quick** | **sky** | **today** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **1** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| **2** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| **3** | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| **4** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
| **5** | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |