**JYOTHY INSTITUTE OF TECHNOLOGY**
**Affiliated to VTU, Belagavi**
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**Accredited by NBA, New Delhi**

# ASSIGNMENT 2

| | |
|---|---|
| Course Code | 18CS71 |
| Course Name | Artificial Intelligence and Machine Learning |

| | |
|---|---|
| USN | 1JT18CS063 |
| Name | Vinyas S |
| Semester | 7 |
| Academic Year | 2021-2022 |

Signature of student　　　　Signature of Instructor

Department of Computer Science and Engineering
Jyothy Institute of Technology, Bangalore – 560 082

# Assignment No. 2 (CO 6)

| | |
|---|---|
| Course Code | 18CS71 |
| Course Name | Artificial Intelligence and Machine Learning |
| Semester | 7 |
| Program | Computer Science and Engineering |

1.What is Stemming? Design and develop a program to demonstrate the working of stemming.

Stemming is the process of producing morphological variants of a root/base word.

Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words.

“chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”

Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words.

Some more example of stemming for root word "like" include:

->"likes"

->"liked"

->"likely"

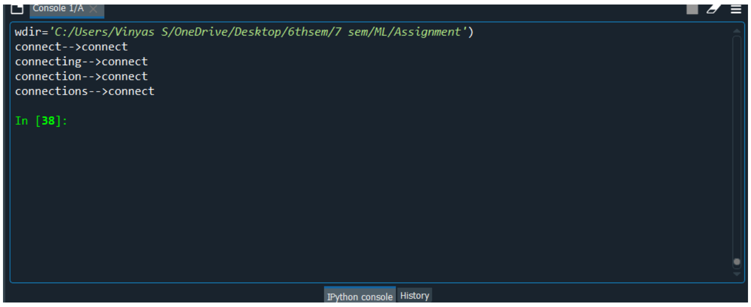
->"liking"

**PROGRAM:**

```
import nltk

from nltk.stem import PorterStemmer

from nltk.tokenize import sent_tokenize,word_tokenize
```

```
ps=PorterStemmer()

tokens=['connect','connecting','connection','connections']


for w in tokens:

    print(w+'-->'+ps.stem(w))
```

**OUTPUT:**



```
wdir='C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment')
connect-->connect
connecting-->connect
connection-->connect
connections-->connect

In [38]:
```

2. What is Lemmatization? Design and develop a program to demonstrate the working of Lemmatization.

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.

Lemmatization is similar to stemming but it brings context to the words.

So it links words with similar meanings to one word.

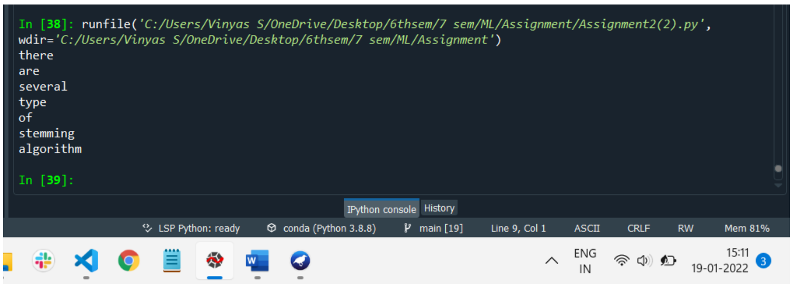
Text preprocessing includes both Stemming as well as Lemmatization

**PROGRAM:**

```
from nltk.stem import WordNetLemmatizer

from nltk.tokenize import sent_tokenize,word_tokenize


lemmatizer=WordNetLemmatizer()

input="there are several types of stemming algorithms"

input=word_tokenize(input)
```

```
for word in input:

    print(lemmatizer.lemmatize(word))
```

**OUTPUT:**



```
In [38]: runfile('C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment/Assignment2(2).py',
wdir='C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment')
there
are
several
type
of
stemming
algorithm

In [39]:
```

3. What is Bag of Words? Design and develop a program to demonstrate the concept of Bag of Words.

The bag-of-words model is a way of representing text data when modeling text with machine learning algorithms.

The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1)A vocabulary of known words.

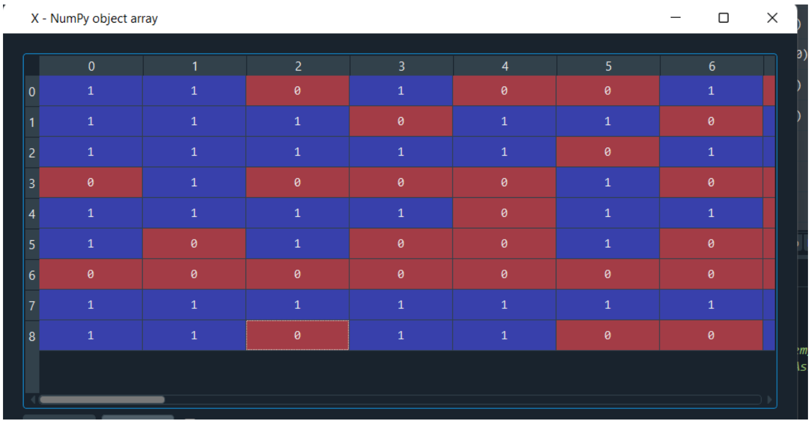
2)A measure of the presence of known words.

**PROGRAM:**

```
import nltk

import re

import numpy as np

import heapq
```

```
text = "Intellectual property validation over the internet for new
content especially in the domain of academia is very robust. But,
they have limitations in forms of not being open source or openly
available, being a too narrow domain or a too broad range problem,
being restricted to only certain formats of data, or many other
ethical concerns that throw out a wide range of exceptional cases
that need to be handled outside the generic checking protocols.
Although text based plagiarism checks available online are prominent
and efficient, the modern output of data on research or anything in
general is more dominantly media content like images, videos, audio
or a combination of these formats of data. Multimedia alone has the
potential to generate revenue. Duplication of content over the
internet is something that is impossible to stop, but if the
validation of the owner or creator of the content is properly
recognised, then it will be much easier to recognise and support the
appropriate content/creator. Such systems are already available as
video copyright check on YouTube or stream copy-strike or copyright
check on Twitch, but they are limited to only their domain of media
i.e. video. To have a general or universal check of content, it is
important to observe a given piece of data in all the formats and
using different perspective learners. This project aims at creating
a copyright infringement checker/ plagiarism tokenizer for
multimedia content making use of, Natural Language, Machine Learning
Processing, Deep Learning, Fuzzy Logic, Big Data and Analytics for
checking the authenticity of the data and Blockchain Technologies,
and Cryptography for assigning a unique identification token for
such content"

dataset = nltk.sent_tokenize(text)

for i in range(len(dataset)):

    dataset[i] = dataset[i].lower()

    dataset[i] = re.sub(r'\W', ' ', dataset[i])

    dataset[i] = re.sub(r'\s+', ' ', dataset[i])

word2count = {}

for data in dataset:

    words = nltk.word_tokenize(data)

    for word in words:

        if word not in word2count.keys():

            word2count[word] = 1
        else:
            word2count[word] += 1

freq_words = heapq.nlargest(100, word2count, key=word2count.get)

X = []
for data in dataset:
    vector = []
    for word in freq_words:
        if word in nltk.word_tokenize(data):
            vector.append(1)
        else:
            vector.append(0)
    X.append(vector)
X = np.asarray(X)
```

**OUTPUT:**



X - NumPy object array

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 5 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

4. What is TF-IDF? Design and develop a program to demonstrate the concept of TF-IDF.

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

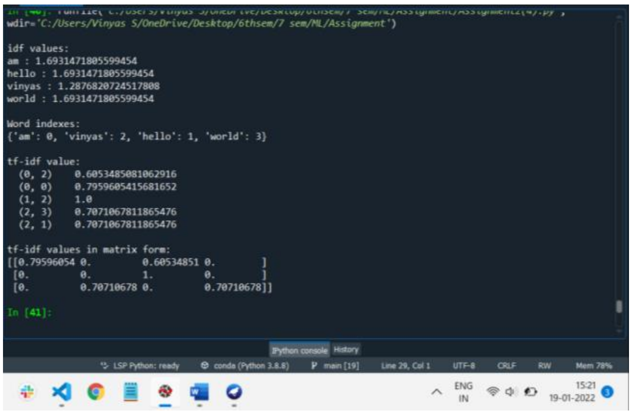
IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

## PROGRAM:

```
from sklearn.feature_extraction.text import TfidfVectorizer

d0 = 'I am Vinyas'

d1 = 'Vinyas'

d2 = 'hello world'

string = [d0, d1, d2]

tfidf = TfidfVectorizer()


result = tfidf.fit_transform(string)
```

```
print('\nidf values:')

for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):

      print(ele1, ':', ele2)


print('\nWord indexes:')

print(tfidf.vocabulary_)


print('\ntf-idf value:')

print(result)


print('\ntf-idf values in matrix form:')

print(result.toarray())
```

**OUTPUT:**



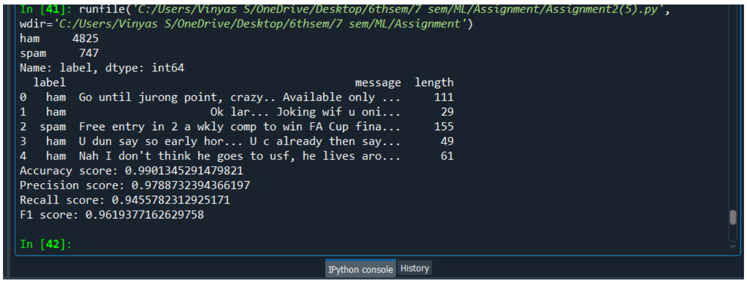
```
In [40]: runfile('C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment/Assignment2(4).py',
wdir='C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment')

idf values:
am : 1.6931471805599454
hello : 1.6931471805599454
vinyas : 1.2876820724517808
world : 1.6931471805599454

Word indexes:
{'am': 0, 'vinyas': 2, 'hello': 1, 'world': 3}

tf-idf value:
  (0, 2)	0.6053485081062916
  (0, 0)	0.7959605415681652
  (1, 2)	1.0
  (2, 3)	0.7071067811865476
  (2, 1)	0.7071067811865476

tf-idf values in matrix form:
[[0.79596054 0.         0.60534851 0.        ]
 [0.         0.         1.         0.        ]
 [0.         0.70710678 0.         0.70710678]]

In [41]:
```

5. Design and develop a program to predict whether a given message is a spam or a ham

PROGRAM:

```
import nltk

from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

import matplotlib.pyplot as plt

import pandas as pd

import string

import seaborn as sns


df = pd.read_csv("smsspamcollection/SMSSpamCollection", sep="\t",
names=["label","message"])

df.head(2)


df = df.rename(columns={"v1":"label", "v2":"message"})

print(df.label.value_counts())


df['length'] = df['message'].apply(len)

print(df.head())


df['length'].plot(bins=50, kind='hist')

df.hist(column='length', by='label', bins=50,figsize=(10,4))

df.loc[:,'label'] = df.label.map({'ham':0, 'spam':1})


X_train, X_test, y_train, y_test = train_test_split(df['message'],


df['label'],test_size=0.20,
                                                    random_state=1)


count_vector = CountVectorizer()


training_data = count_vector.fit_transform(X_train)


testing_data = count_vector.transform(X_test)


naive_bayes = MultinomialNB()

naive_bayes.fit(training_data,y_train)


predictions = naive_bayes.predict(testing_data)

print('Accuracy score: {}'.format(accuracy_score(y_test,
predictions)))

print('Precision score: {}'.format(precision_score(y_test,
predictions)))

print('Recall score: {}'.format(recall_score(y_test, predictions)))

print('F1 score: {}'.format(f1_score(y_test, predictions)))
```

OUTPUT:



```
In [41]: runfile('C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment/Assignment2(5).py',
wdir='C:/Users/Vinyas S/OneDrive/Desktop/6thsem/7 sem/ML/Assignment')
ham     4825
spam     747
Name: label, dtype: int64
  label                                            message  length
0   ham  Go until jurong point, crazy.. Available only ...     111
1   ham                      Ok lar... Joking wif u oni...      29
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...     155
3   ham  U dun say so early hor... U c already then say...      49
4   ham  Nah I don't think he goes to usf, he lives aro...      61
Accuracy score: 0.9901345291479821
Precision score: 0.9788732394366197
Recall score: 0.9455782312925171
F1 score: 0.9619377162629758

In [42]:
```

Department of Computer Science and Engineering
Jyothy Institute of Technology, Bangalore – 560 082