CSE6060

Statistical Natural Language Processing

Activity 1

Name : Kavianand G

Reg. No. : 19MAI0050

Date : 13 – June – 2020

Cosine Similarity

In [1]:

```
data_1 = "Data is the oil of the digital economy"
data_2 = "Data is a new oil"

data = [data_1, data_2]
```

Using CountVectorizer

In [2]:

```
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
vector_matrix = count_vect.fit_transform(data)
print(vector_matrix)
```

```
  (0, 0)	1
  (0, 3)	1
  (0, 7)	2
  (0, 6)	1
  (0, 5)	1
  (0, 1)	1
  (0, 2)	1
  (1, 0)	1
  (1, 3)	1
  (1, 6)	1
  (1, 4)	1
```

In [3]:

```
tokens = count_vect.get_feature_names()
print(tokens)
```

```
['data', 'digital', 'economy', 'is', 'new', 'of', 'oil', 'the']
```

In [4]:

```
vocab = count_vect.vocabulary_
vocab
```

Out[4]:

```
{'data': 0,
 'is': 3,
 'the': 7,
 'oil': 6,
 'of': 5,
 'digital': 1,
 'economy': 2,
 'new': 4}
```

In [5]:

```
vec_data_1 = count_vect.transform([data_1]).toarray()
print(vec_data_1)
```

```
[[1 1 1 1 0 1 1 2]]
```

In [6]:

```
vec_data_2 = count_vect.transform([data_2]).toarray()
print(vec_data_2)
```

```
[[1 0 0 1 1 0 1 0]]
```

In [7]:

```
matrix = vector_matrix.toarray()
print(matrix)
```

```
[[1 1 1 1 0 1 1 2]
 [1 0 0 1 1 0 1 0]]
```

In [8]:

```
import pandas as pd

def create_dataframe(matrix, tokens):

    doc_names = [f'doc_{i+1}' for i, _ in enumerate(matrix)]
    df = pd.DataFrame(data=matrix, index=doc_names, columns=tokens)
    return(df)
```

In [9]:

```
create_dataframe(matrix,tokens)
```

Out[9]:

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

In [10]:

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity_matrix = cosine_similarity(vector_matrix)
create_dataframe(cosine_similarity_matrix,['doc_1','doc_2'])
```

Out[10]:

	doc_1	doc_2
doc_1	1.000000	0.474342
doc_2	0.474342	1.000000

In [11]:

```
print("Cosine Similarity = ",(cosine_similarity(vec_data_1,vec_data_2))[0][0])
```

```
Cosine Similarity =  0.4743416490252569
```

Using TfidfVectorizer

In [12]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

Tfidf_vect = TfidfVectorizer()
vector_matrix = Tfidf_vect.fit_transform(data)

tokens = Tfidf_vect.get_feature_names()
create_dataframe(vector_matrix.toarray(),tokens)
```

Out[12]:

	data	digital	economy	is	new	of	oil	the
doc_1	0.243777	0.34262	0.34262	0.243777	0.000000	0.34262	0.243777	0.68524
doc_2	0.448321	0.00000	0.00000	0.448321	0.630099	0.00000	0.448321	0.00000

In [13]:

```
cosine_similarity_matrix = cosine_similarity(vector_matrix)
create_dataframe(cosine_similarity_matrix,['doc_1','doc_2'])
```

Out[13]:

	doc_1	doc_2
doc_1	1.000000	0.327871
doc_2	0.327871	1.000000

In [14]:

```
vec_data_1 = Tfidf_vect.transform([data_1]).toarray()
print(vec_data_1)
```

```
[[0.24377685 0.34261985 0.34261985 0.24377685 0.         0.34261985
  0.24377685 0.68523971]]
```

In [15]:

```
vec_data_2 = Tfidf_vect.transform([data_2]).toarray()
print(vec_data_2)
```

```
[[0.44832087 0.         0.         0.44832087 0.63009934 0.
  0.44832087 0.        ]]
```

In [16]:

```
print("Cosine Similarity = ",(cosine_similarity(vec_data_1,vec_data_2))[0][0])
```

```
Cosine Similarity =  0.3278707471841718
```

Simple Text Classifier

In [17]:

```
from nltk.corpus import names
import random
```

In [18]:

```
male_name =[(name, 'male') for name in names.words('male.txt')]
female_name = [(name, 'female') for name in names.words('female.txt')]
```

In [19]:

```
print(male_name,female_name)
```

```
[('Aamir', 'male'), ('Aaron', 'male'), ('Abbey', 'male'), ('Abbie', 'mal
e'), ('Abbot', 'male'), ('Abbott', 'male'), ('Abby', 'male'), ('Abdel', 'm
ale'), ('Abdul', 'male'), ('Abdulkarim', 'male'), ('Abdullah', 'male'),
('Abe', 'male'), ('Abel', 'male'), ('Abelard', 'male'), ('Abner', 'male'),
('Abraham', 'male'), ('Abram', 'male'), ('Ace', 'male'), ('Adair', 'mal
e'), ('Adam', 'male'), ('Adams', 'male'), ('Addie', 'male'), ('Adger', 'ma
le'), ('Aditya', 'male'), ('Adlai', 'male'), ('Adnan', 'male'), ('Adolf',
'male'), ('Adolfo', 'male'), ('Adolph', 'male'), ('Adolphe', 'male'), ('Ad
olpho', 'male'), ('Adolphus', 'male'), ('Adrian', 'male'), ('Adrick', 'mal
e'), ('Adrien', 'male'), ('Agamemnon', 'male'), ('Aguinaldo', 'male'), ('A
guste', 'male'), ('Agustin', 'male'), ('Aharon', 'male'), ('Ahmad', 'mal
e'), ('Ahmed', 'male'), ('Ahmet', 'male'), ('Ajai', 'male'), ('Ajay', 'mal
e'), ('Al', 'male'), ('Alaa', 'male'), ('Alain', 'male'), ('Alan', 'mal
e'), ('Alasdair', 'male'), ('Alastair', 'male'), ('Albatros', 'male'), ('A
lbert', 'male'), ('Alberto', 'male'), ('Albrecht', 'male'), ('Alden', 'mal
e'), ('Aldis', 'male'), ('Aldo', 'male'), ('Aldric', 'male'), ('Aldrich',
'male'), ('Aldus', 'male'), ('Aldwin', 'male'), ('Alec', 'male'), ('Alec
k', 'male'), ('Alejandro', 'male'), ('Aleks', 'male'), ('Aleksandrs', 'mal
e'), ('Alessandro', 'male'), ('Alex', 'male'), ('Alexander', 'male'), ('Al
```

In [20]:

```
labelled_name = male_name + female_name
random.shuffle(labelled_name)
```

In [21]:

```
print(labelled_name)
```

```
[('Angie', 'female'), ('Almeta', 'female'), ('Laure', 'female'), ('Nyssa',
'female'), ('Jared', 'male'), ('Fletcher', 'male'), ('Florina', 'female'),
('Chip', 'male'), ('Kayle', 'female'), ('Josey', 'female'), ('Rhona', 'fem
ale'), ('Walter', 'male'), ('Simonette', 'female'), ('Fionnula', 'femal
e'), ('Nico', 'male'), ('Giacinta', 'female'), ('Val', 'male'), ('Lauree
n', 'female'), ('Edie', 'male'), ('Shalom', 'male'), ('Corby', 'male'),
('Clarey', 'female'), ('Ellene', 'female'), ('Elliott', 'male'), ('Elga',
'female'), ('Lefty', 'male'), ('Ursa', 'female'), ('Wilone', 'female'),
('Tamas', 'male'), ('Clemmie', 'female'), ('Mareah', 'female'), ('Ruthi',
'female'), ('Murphy', 'male'), ('Arnie', 'male'), ('Cariotta', 'female'),
('Klarrisa', 'female'), ('Munroe', 'male'), ('Anne-Mar', 'female'), ('Nath
anial', 'male'), ('Janel', 'female'), ('Todd', 'male'), ('Legra', 'femal
e'), ('Robbyn', 'female'), ('Fatima', 'female'), ('Pieter', 'male'), ('Bil
li', 'female'), ('Chrissy', 'male'), ('Zak', 'male'), ('Giralda', 'femal
e'), ('Goldy', 'female'), ('Casey', 'female'), ('Koral', 'female'), ('Nanc
ie', 'female'), ('Cristie', 'female'), ('Abbie', 'female'), ('Gustavo', 'm
ale'), ('Valene', 'female'), ('Tiffanie', 'female'), ('Max', 'male'), ('Ro
ni', 'male'), ('Mika', 'male'), ('Nahum', 'male'), ('Carmella', 'female'),
('Constantinos', 'male'), ('Hammad', 'male'), ('Blondie', 'female'), ('Els
```

In [22]:

```
print(len(labelled_name))
```

```
7944
```

In [23]:

```
def gender_features(word): #gives last letter of the word
    return {'last_letter':word[-1]}
```

In [24]:

```
featuresets = [(gender_features(n),gender) for (n,gender) in labelled_name]
```

In [25]:

```
featuresets
```

```
 ({'last_letter': 'l'}, 'female'),
 ({'last_letter': 'd'}, 'male'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'n'}, 'female'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'r'}, 'male'),
 ({'last_letter': 'i'}, 'female'),
 ({'last_letter': 'y'}, 'male'),
 ({'last_letter': 'k'}, 'male'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'y'}, 'female'),
 ({'last_letter': 'y'}, 'female'),
 ({'last_letter': 'l'}, 'female'),
 ({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'o'}, 'male'),
 ({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'e'}, 'female'),
```

In [26]:

```
train_set, test_set = featuresets[500:], featuresets[:500]
```

In [27]:

```
print(len(train_set),len(test_set))
```

```
7444 500
```

In [28]:

```
import nltk
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [29]:

```
train_set_acc = nltk.classify.accuracy(classifier, train_set)
test_set_acc = nltk.classify.accuracy(classifier, test_set)
```

In [30]:

```
print("Accuracy on Train dataset = ", train_set_acc)
print("Accuracy on Test dataset = ", test_set_acc)
```

```
Accuracy on Train dataset =  0.7614185921547555
Accuracy on Test dataset =  0.784
```

In [31]:

```
classifier.classify(gender_features("Kavianand"))
```

Out[31]:

```
'male'
```

In [32]:

```
classifier.classify(gender_features("Kavi"))
```

Out[32]:

```
'female'
```

In [33]:

```
classifier.classify(gender_features("Kavin"))
```

Out[33]:

```
'male'
```

In [34]:

```
classifier.classify(gender_features("Rose"))
```

Out[34]:

```
'female'
```

---End of Documentation---