One-hot encoding and Bag of Words

Toy Corpus

```
d = []
d.append("I am studying Natural language processing at Amrita Amrita")
d.append("Amrita Amrita offers Natural Language Processing Processing")
d.append("Natural language processing is offered at Amrita")
d.append("It was raining yesterday")
d.append("Yesterday I learned Natural language processing in the rain")
```

```
##clean data() Pre-process
vocabulary = []
for doc in d:
    for w in doc.split():
        if w.lower() not in vocabulary:
            vocabulary.append(w.lower())
vocabulary
```

```
['i',
 'am',
 'studying',
 'natural',
 'language',
 'processing',
 'at',
 'amrita',
 'offers',
 'is',
 'offered',
 'it',
 'was',
 'raining',
 'yesterday',
 'learned',
 'in',
 'the',
 'rain']
```

```
vocabulary.sort()
vocabulary
```

```
['am',
 'amrita',
 'at',
 'i',
 'in',
```

```
 'is',
 'it',
 'language',
 'learned',
 'natural',
 'offered',
 'offers',
 'processing',
 'rain',
 'raining',
 'studying',
 'the',
 'was',
 'yesterday']
```

```
import numpy as np
```

```
word_vectors = []
i = 0
for w in vocabulary:
    t = np.zeros(len(vocabulary), dtype = np.int16)
    t[i] = 1
    word_vectors.append(t)
    i = i + 1
i=0
dic={}
for e in word_vectors :
    print(vocabulary[i],'-->', e)
    i=i+1
```

```
 am --> [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 amrita --> [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 at --> [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 i --> [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 in --> [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 is --> [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 it --> [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 language --> [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 learned --> [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 natural --> [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 offered --> [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 offers --> [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 processing --> [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 rain --> [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 raining --> [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 studying --> [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 the --> [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 was --> [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 yesterday --> [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
text = "Amrita is studying Natural Language Processing"
```

```
text=text.lower()
text
```

```
'amrita is studying natural language processing'
```

```
tokens = text.split()

tokens
```

```
['amrita', 'is', 'studying', 'natural', 'language', 'processing']
```

```
doc_V=[]
for e in tokens:
  doc_V.append(word_vectors[vocabulary.index(e)])
```

```
import pandas as pd
df=pd.DataFrame(doc_V)
df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

```
df.columns=vocabulary
df.index=tokens
df
```

	am	amrita	at	i	in	is	it	language	learned	natural	offered	offer
amrita	0	1	0	0	0	0	0	0	0	0	0	
is	0	0	0	0	0	1	0	0	0	0	0	
studying	0	0	0	0	0	0	0	0	0	0	0	
natural	0	0	0	0	0	0	0	0	0	1	0	
language	0	0	0	0	0	0	0	1	0	0	0	
processing	0	0	0	0	0	0	0	0	0	0	0	

pd.dummies()

```
import pandas as pd
pd.get_dummies(vocabulary)
```

	am	amrita	at	i	in	is	it	language	learned	natural	offered	offers	proce
0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	0	0	
3	0	0	0	1	0	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	0	0	0	
5	0	0	0	0	0	1	0	0	0	0	0	0	
6	0	0	0	0	0	0	1	0	0	0	0	0	
7	0	0	0	0	0	0	0	1	0	0	0	0	
8	0	0	0	0	0	0	0	0	1	0	0	0	
9	0	0	0	0	0	0	0	0	0	1	0	0	
10	0	0	0	0	0	0	0	0	0	0	1	0	
11	0	0	0	0	0	0	0	0	0	0	0	1	
12	0	0	0	0	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	

BagofWords

```
docV = []
```

```
for doc in d:
    dt = np.zeros(len(vocabulary), dtype = np.int16)
    for w in doc.split():
        if w.lower() in vocabulary:
            indexOfW = vocabulary.index(w.lower())
```

```
            dt[indexOfW] +=1
    docV.append(dt)
```

d = [] d.append("I am studying Natural language processing at Amrita")

d.append("Amrita offers Natural Language Processing")

d.append("Natural language processing is offered at Amrita")

d.append("It was raining yesterday")

d.append("Yesterday I learned Natural language processing in the rain")

```
for d in docV:
    print(d)
```

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

```
df2=pd.DataFrame(docV)
df2.columns=vocabulary
df2
```

	am	amrita	at	i	in	is	it	language	learned	natural	offered	offers	proces
0	1	2	1	1	0	0	0	1	0	1	0	0	
1	0	2	0	0	0	0	0	1	0	1	0	1	
2	0	1	1	0	0	1	0	1	0	1	1	0	
3	0	0	0	0	0	0	1	0	0	0	0	0	
4	0	0	0	1	1	0	0	1	1	1	0	0	

Similarity Measures

```
import math
```

```
def cosSimilar(v1,v2):

    sum = 0
    lv1 = 0
    lv2 = 0
    for i in range(v1.shape[0]):
        sum += v1[i]*v2[i]
```

```
        lv1 += v1[i]*v1[i]
        lv2 += v2[i]*v2[i]
    dp = sum / (math.sqrt(lv1) * math.sqrt(lv2))
    return dp
```

```
def cosSimilar2(A,B):
    res = np.dot(A,B)/(np.linalg.norm(A)*np.linalg.norm(B))
    return res
```

```
A=np.array([7,7])
B=np.array([7,-7])
cosSimilar2(A,B)
```

```
0.0
```

```
cosSimilar(A,B)
```

```
0.0
```

```
cosSimilar(docV[1],docV[2])
```

```
0.6837634587578276
```

```
cosSimilar(docV[0],docV[4])
```

```
0.40201512610368484
```

```
cosSimilar2(docV[0],docV[4])
```

```
0.40201512610368484
```

```
#euclidean distance
```

```
def euclSimilarity(v1,v2):
    sum = 0
    for i in range(v1.shape[0]):
        sum += (v1[i]-v2[i])*(v1[i]-v2[i])
    res=1/(1+math.sqrt(sum))
    #res=math.sqrt(sum)
    return res
```

```
euclSimilarity(A,B)
```

```
0.06666666666666667
```

```
cosSimilar2(docV[1],docV[1])
```

```
1.0
```

```
euclSimilarity(docV[1],docV[1])
```

```
1.0
```

```
def euclSimilarity2(v1,v2):
    res = np.sqrt(np.sum((v1 - v2) ** 2))
    res = 1/(1+res)
    return res
```

```
euclSimilarity2(docV[0],docV[1])
```

```
0.28989794855663564
```

```
def textToVector(text):
    vec = np.zeros(len(vocabulary), dtype = np.int16)
    for w in text.split():
        if w.lower() in vocabulary:
            indexOfW = vocabulary.index(w.lower())
            vec[indexOfW] +=1
    return vec
```

```
text = "Amrita is studying Natural Language Processing"
tv = textToVector(text)
tv
```

```
array([0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0],
      dtype=int16)
```

```
df3=pd.DataFrame(tv)
df3.index = vocabulary
df3.transpose()
```

	am	amrita	at	i	in	is	it	language	learned	natural	offered	offers	proces
0	0	1	0	0	0	1	0	1	0	1	0	0	

```
for d in docV:
    print(cosSimilar2(d,tv))
```

```
0.7385489458759965
0.7385489458759965
0.7715167498104595
0.0
0.4082482904638631
```

```
for d in docV:
    print(euclSimilarity2(d,tv))
```

```
0.3090169943749474
0.3090169943749474
0.36602540378443865
0.2402530733520421
0.25
```

```

```