:[1] In

```
from __future__ import print_function
from nltk.metrics import *

Sentence1='There are many similarity measures used in NLTK package'.split()
Sentence2='There are many similarity measures are avaliable in NLTK '.split()

print('Accuracy = ',accuracy(Sentence1,Sentence2))
```

```
Accuracy =  0.5555555555555556
```

:[2] In

```
setSentence1=set(Sentence1)
setSentence2=set(Sentence2)

precision = precision(setSentence1,setSentence2)
recall = recall(setSentence1,setSentence2)

print('Precision = ',precision)
print('Recall = ',recall)
```

```
Precision =  0.875
Recall =  0.7777777777777778
```

:[3] In

```
f_measure = (2 * precision * recall) / (precision + recall)
print('F-measure = ',f_measure)
```

```
F-measure =  0.823529411764706
```

:[4] In

```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report

confusion_matrix = confusion_matrix(Sentence1, Sentence2)

print('Confusion Matrix \n', confusion_matrix)
```

```
 Confusion Matrix
[0 0 0 0 0 1 0 0 0 0]]
[0 0 0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0]
[0 0 0 0 1 0 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1]
[0 1 0 0 0 0 0 0 0 0]
[[0 0 0 0 0 0 0 1 0 0]
```

:[5] In

```
classification_report = classification_report(Sentence1, Sentence2)
print('Classification Report \n', classification_report)
```

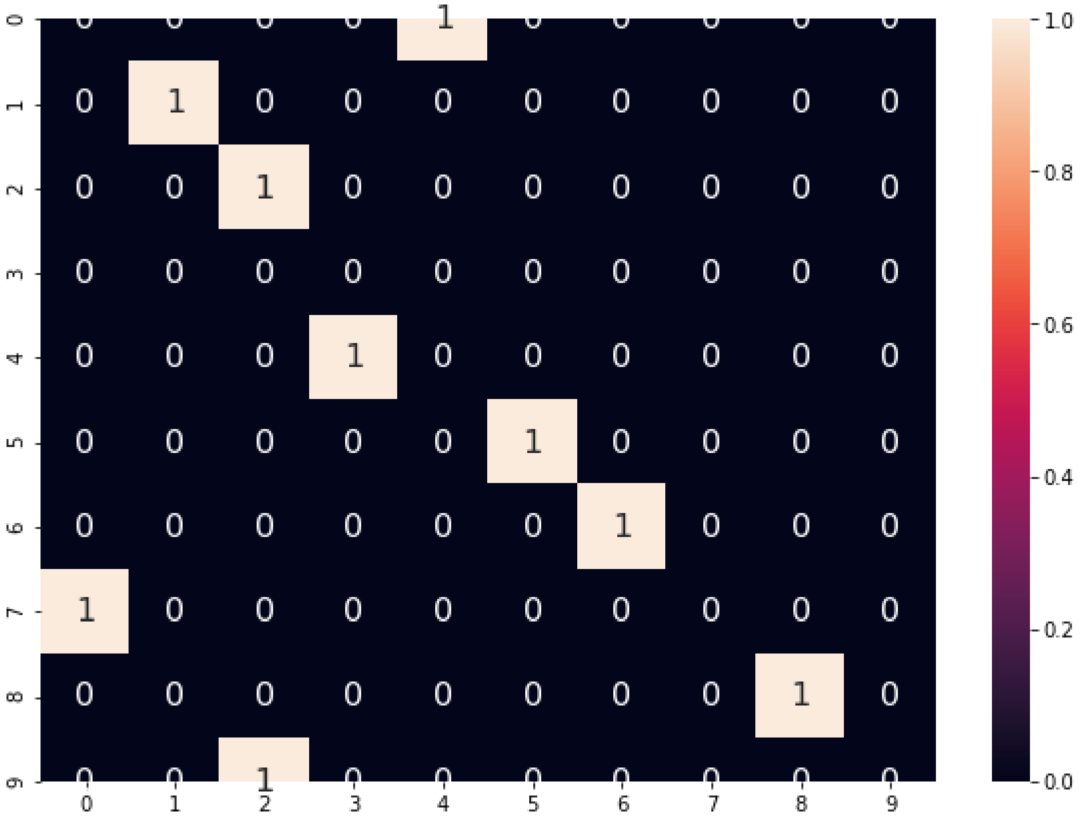
```
 Classification Report
precision    recall  f1-score   support

NLTK       0.00      0.00      0.00         1
There       1.00      1.00      1.00         1
are       0.50      1.00      0.67         1
avaliable       0.00      0.00      0.00         0
in       0.00      0.00      0.00         1
many       1.00      1.00      1.00         1
measures       1.00      1.00      1.00         1
package       0.00      0.00      0.00         1
similarity       1.00      1.00      1.00         1
used       0.00      0.00      0.00         1

accuracy                           0.56         9
macro avg       0.45      0.50      0.47         9
weighted avg       0.50      0.56      0.52         9


D:\Users\D7me_\Anaconda3\lib\site-packages\sklearn\metrics\classification.p
y:1437: UndefinedMetricWarning: Precision and F-score are ill-defined and be
.ing set to 0.0 in labels with no predicted samples
(precision', 'predicted', average, warn_for'
D:\Users\D7me_\Anaconda3\lib\site-packages\sklearn\metrics\classification.p
 y:1439: UndefinedMetricWarning: Recall and F-score are ill-defined and being
.set to 0.0 in labels with no true samples
(recall', 'true', average, warn_for'
```

:[6] In

```
df_cm = pd.DataFrame(confusion_matrix)
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True, annot_kws={"size": 16})
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f0752da7c8>
```



:[7] In

```
import nltk
from nltk.metrics import *
print(edit_distance("relate","relation"))
print(edit_distance("suggestion","calculation"))
```

```
3
7
```

:[8] In

```
import nltk
from nltk.metrics import *

def jacc_similarity(query, document):
    first=set(query).intersection(set(document))
    second=set(query).union(set(document))
    return len(first)/len(second)

X = set(Sentence1)
Y = set(Sentence2)

print(jaccard_distance(X,Y))
```

```
0.3
```

:[9] In

```
def binary_distance(label1, label2):
    return 0.0 if label1 == label2 else 1.0

X=set(Sentence1)
Y=set(Sentence2)
binary_distance(X, Y)
```

Out[9]:

```
1.0
```

:[10] In

```
def masi(label1, label2):
    len_intersection = len(label1.intersection(label2))
    len_union = len(label1.union(label2))
    len_label1 = len(label1)
    len_label2 = len(label2)
    if len_label1 == len_label2 and len_label1 == len_intersection:
        m = 1
    elif len_intersection == min(len_label1, len_label2):
        m = 0.67
    elif len_intersection > 0:
        m = 0.33
    else:
        m = 0
    return 1 - (len_intersection / float(len_union)) * m

X=set([10,20,30,40])
Y=set([30,50,70])
masi(X, Y)
```

Out[10]:

```
0.945
```

EX: 3

:[11] In

```
import nltk
from nltk.metrics import *

def jacc_similarity(query, document):
    first=set(query).intersection(set(document))
    second=set(query).union(set(document))
    return len(first)/len(second)

file1 = open(r'D:\Users\D7me_\Anaconda3\Abdulrhman.txt').read().split()
file2 = open(r'D:\Users\D7me_\Anaconda3\Abdulrhman.txt').read().split()

text1 = set(file1)
text2 = set(file2)

print('text1\n',text1)
print('text2\n',text2)

print('\n\nSimilarity =', jaccard_distance(text1,text2))
```

```
text1
{'B777-300ER'}
text2
{'B777-300ER'}


Similarity = 0.0
```