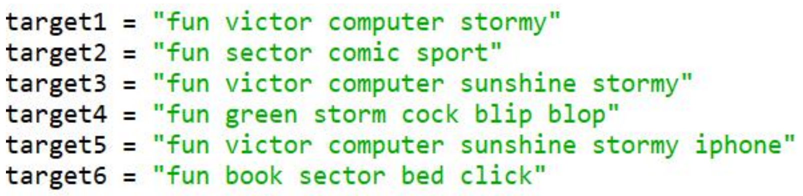
BINGBING LI Text Analytics COMP47600 20.10.2019
18210705 Practical 5

Q1：Make up your own set of word features describing 6 different entities; with some obvious overlaps and differences.

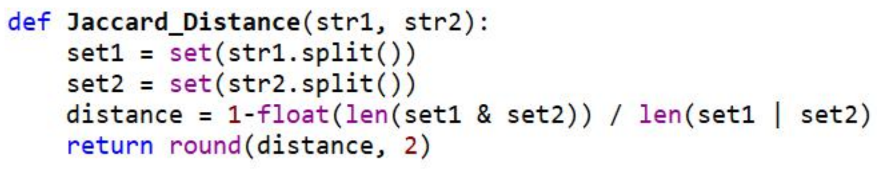
My text is:



```
target1 = "fun victor computer stormy"
target2 = "fun sector comic sport"
target3 = "fun victor computer sunshine stormy"
target4 = "fun green storm cock blip blop"
target5 = "fun victor computer sunshine stormy iphone"
target6 = "fun book sector bed click"
```

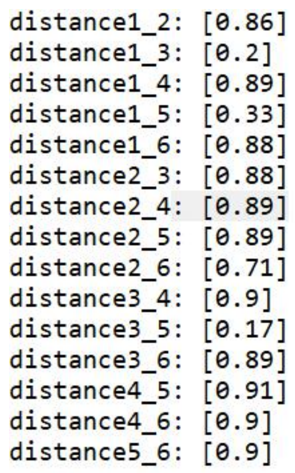
Q1(a): Modify the Jaccard-Index python program to do Jaccard-Distance and then compute all pairwise distances between the entities. Based on results, show empirically, that the property of triangle inequality holds for measure.

The Jaccard-Distance method after modified:



```
def Jaccard_Distance(str1, str2):
    set1 = set(str1.split())
    set2 = set(str2.split())
    distance = 1-float(len(set1 & set2)) / len(set1 | set2)
    return round(distance, 2)
```

The Jaccard distances between these six entities:



```
distance1_2: [0.86]
distance1_3: [0.2]
distance1_4: [0.89]
distance1_5: [0.33]
distance1_6: [0.88]
distance2_3: [0.88]
distance2_4: [0.89]
distance2_5: [0.89]
distance2_6: [0.71]
distance3_4: [0.9]
distance3_5: [0.17]
distance3_6: [0.89]
distance4_5: [0.91]
distance4_6: [0.9]
distance5_6: [0.9]
```

Based on the results,it can be proved that the property of triangle inequality holds for measure. For instance,

(distance1_2 + distance1_3) > distance2_3,

(distance1_2 - distance1_3) < distance2_3

Furthermore every three triangular distances achieve this rule.

Q1(b): Now implement the difference function for the Dice Coefficient and show that the property of triangle inequality may not hold for this measure.

Dice Coefficient method:

```
def DiceCoefficient(str1,str2):
    set1 = set(str1.split())
    set2 = set(str2.split())
    distance = 1-2*float(len(set1 & set2)) / (len(set1)+len(set2))
    return round(distance, 2)
```

The results about distance using Dice Coefficient method:

```
1_2 [0.75]
1_3 [0.11]
1_4 [0.8]
1_5 [0.2]
1_6 [0.78]
2_3 [0.78]
2_4 [0.8]
2_5 [0.8]
2_6 [0.56]
3_4 [0.82]
3_5 [0.09]
3_6 [0.8]
4_5 [0.83]
4_6 [0.82]
5_6 [0.82]
```

Based on the result, it can be proved that the property of triangle inequality may not hold for this measure. For instance:

(Distance1_3 + distance3_5) = distance1_5

So, triangle inequality may not hold for this measure.

Q2: Have a look at the Cosine.py program; nb you may need to install the packages its imports.

Q2(a): Find 3 short documents about which you might want to know their similarity. Produce 5 variants on one of the documents and see how the cosine similarity changes.

My 3 short documents:

```
doc1=('d1', 'transport network development and its accessibility')
doc2=('d2', 'indicators of megaregional transport network')
doc3=('d3', 'county accessibility to megaregional activities')
```

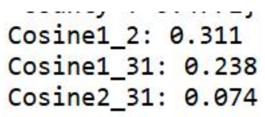
Cosine similarity:

```
Cosine1_2: 0.193
Cosine1_3: 0.076
Cosine2_3: 0.076
```

1. After the first change to doc3:

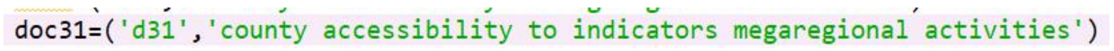
```
doc31=('d31','county accessibility to development megaregional activities')
```

Cosine similarity:



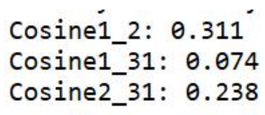
```
Cosine1_2: 0.311
Cosine1_31: 0.238
Cosine2_31: 0.074
```

2. After the second change to doc3:



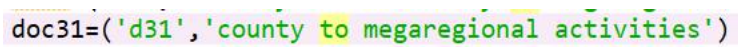
```
doc31=('d31','county accessibility to indicators megaregional activities')
```

Cosine similarity:



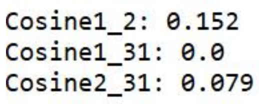
```
Cosine1_2: 0.311
Cosine1_31: 0.074
Cosine2_31: 0.238
```

3. After the third change to doc3:



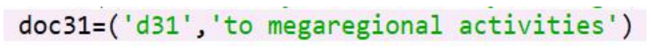
```
doc31=('d31','county to megaregional activities')
```

Cosine similarity:



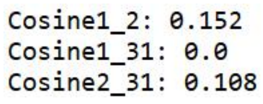
```
Cosine1_2: 0.152
Cosine1_31: 0.0
Cosine2_31: 0.079
```

4. After the forth change to doc3:



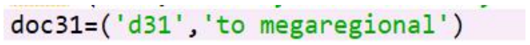
```
doc31=('d31','to megaregional activities')
```

Cosine similarity:



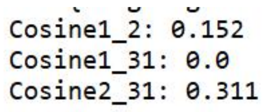
```
Cosine1_2: 0.152
Cosine1_31: 0.0
Cosine2_31: 0.108
```

5. After the fifth change to doc3:



```
doc31=('d31','to megaregional')
```

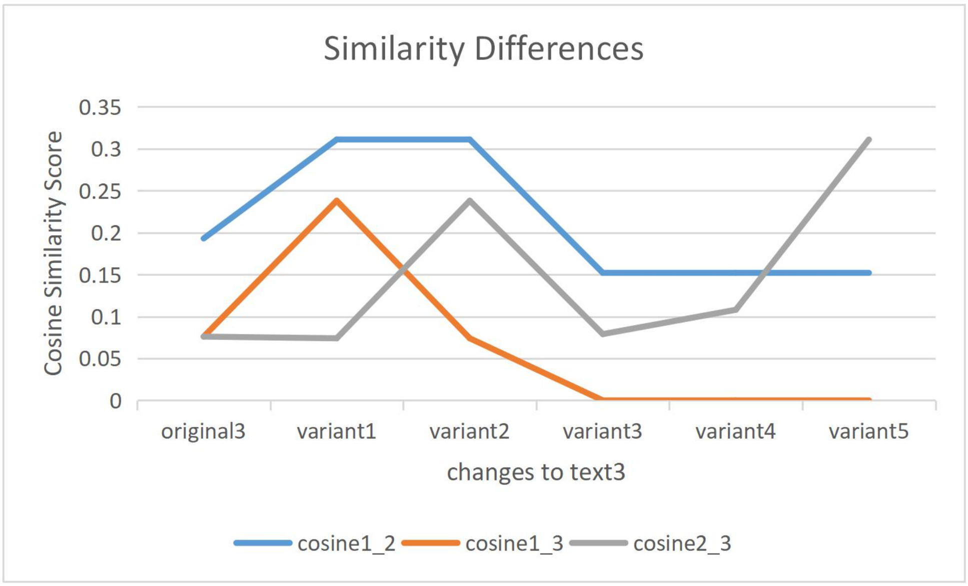
Cosine similarity:



```
Cosine1_2: 0.152
Cosine1_31: 0.0
Cosine2_31: 0.311
```

It can be seen from the result that, The bigger the cosine the smaller the angle the more similar are the vectors. Furthermore when the text3 is changed and the all_words will be changed so it will change the tf_idf scores, so the cosine similarity between text1 and text2 will be changed even if the content of them is intact.

Q2(b): Plot the similarity differences on a graph showing their cosine similarity score. Verify that your intuitions about what makes the differing docs less similar does indeed lead to scores that are less similar.



It can be seen that from variant3, after I change the doc3 to make it more similar to doc2 and less similar to doc1. The Cosine similarity scores shows that the cosine2_3 keeps increasing and the cosine1_3 becomes 0.

Q2(c): Find a python package that computes cosine similarity and euclidean distance. Use it process the data you have already. Do the answers for Cosine Similarity correspond? What do the Euclidean Distance scores look like relative to the Cosine ones?

```
doc1 = 'transport network development and its accessibility'
doc2 = 'indicators of megaregional transport network'
doc3 = 'county accessibility to megaregional activities'
doc3_01 = 'county accessibility to development megaregional activities'
doc3_02 = 'county accessibility to indicators megaregional activities'
doc3_03 = 'county to megaregional activities'
doc3_04 = 'to megaregional activities'
doc3_05 = 'to megaregional'
```

Euclidean Distance scores:

```
[[0.         2.64575131 3.         2.82842712 3.16227766 3.16227766
  3.         2.82842712]
 [2.64575131 0.         2.82842712 3.         2.64575131 2.64575131
  2.44948974 2.23606798]
 [3.         2.82842712 0.         1.         1.         1.
  1.41421356 1.73205081]
 [2.82842712 3.         1.         0.         1.41421356 1.41421356
  1.73205081 2.        ]
 [3.16227766 2.64575131 1.         1.41421356 0.         1.41421356
  1.73205081 2.        ]
 [3.16227766 2.64575131 1.         1.41421356 1.41421356 0.
  1.         1.41421356]
 [3.         2.44948974 1.41421356 1.73205081 1.73205081 1.
  0.         1.        ]
 [2.82842712 2.23606798 1.73205081 2.         2.         1.41421356
  1.         0.        ]]
```

It can be seen that answers don’t correspond to Cosine Similarity .As for Euclidean Distance scores, the bigger the data is, the less similar the texts are.

Q3: Create or find 5 “normal” tweets from Twitter. Now take one of these tweets and sysmatically generate 20 SPAM tweets from it; using the typical techniques of spammers.

Now, perform comparisons between these 20 SPAM tweets each of the 5 Normal Tweets. Plot their edit-distance scores in a graph and colour code to show how the SPAM v Normal ones. Are the SPAM tweets obvious, if not why?

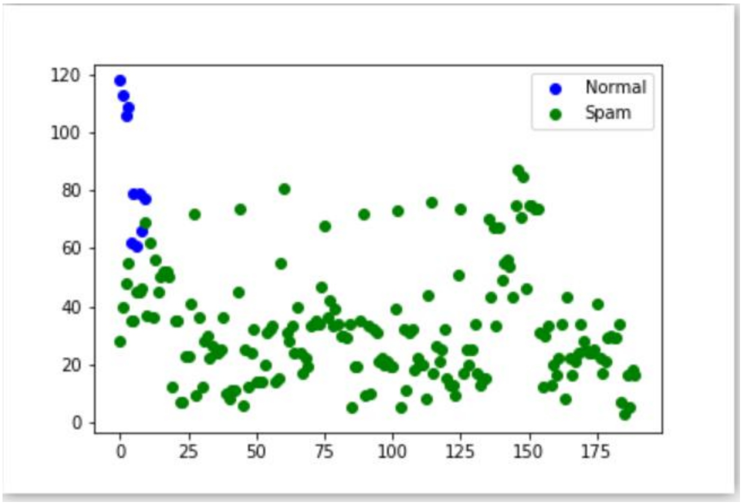
The edit-distance scores of normal tweets:

```
[118, 113, 106, 109, 62, 79, 61, 79, 66, 77]
```

The edit-distance scores of spam tweets:

```
 [28, 40, 48, 55, 35, 35, 45, 45, 46, 69, 37, 62, 36, 56, 45,
50, 52, 52, 50, 12, 35, 35, 7, 7, 23, 23, 41, 72, 9, 36, 12,
28, 30, 22, 26, 24, 24, 25, 36, 10, 8, 11, 11, 45, 74, 6, 25,
12, 24, 32, 14, 14, 14, 20, 31, 32, 33, 14, 15, 55, 81, 31,
28, 33, 24, 40, 24, 17, 22, 19, 33, 34, 35, 34, 47, 68, 36,
42, 33, 39, 34, 30, 30, 29, 34, 5, 19, 19, 35, 72, 9, 33, 10,
32, 31, 21, 22, 20, 21, 19, 19, 39, 73, 5, 32, 11, 31, 32, 18,
22, 20, 20, 8, 44, 76, 17, 26, 21, 25, 32, 15, 13, 13, 9, 51,
74, 17, 25, 20, 25, 34, 17, 13, 15, 15, 70, 43, 67, 33, 67,
49, 55, 56, 54, 43, 75, 87, 71, 85, 46, 75, 75, 74, 74, 31,
12, 30, 33, 13, 20, 16, 22, 34, 8, 43, 22, 16, 21, 24, 34, 28,
25, 24, 24, 25, 41, 22, 17, 21, 29, 30, 29, 29, 34, 7, 3, 16,
5, 18, 16]
```

Graph:



It can be seen from the values that the spam tweets have relatively low edit-distance than normal ones because they have a lot features in common such as http:// and # .

In the graph the green dots represent the spam tweets and the blue dots represent the normal ones.

We can see that the spam tweets are really obvious.