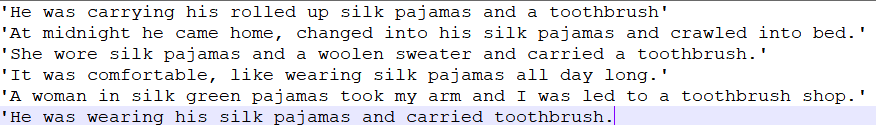
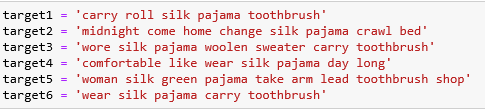
**Q1. Make up your own set of word features describing 6 different entities; with some obvious overlaps and differences.**

Set of word features describing 6 various entities with some overlaps and differences are made up in a data structure dictionary and pre processing is done which includes ( Tokenization, Normalization(Downcase, Stopwords removal and punctuation marks removal), Lemmatization):



Processed set of word features:

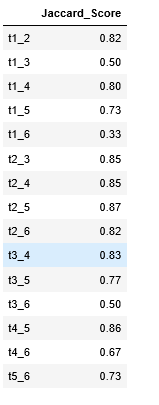


1. **Modify the Jaccard-Index python program to do Jaccard-Distance and then compute all pairwise distances between the entities. Based on results, show empirically, that the property of triangle inequality holds for measure.**

**Ans1a.**

* **Jaccard Index:** It is a statistic used to measure the similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets: J(A,B) = |A & B|/|A|B|.
* **Jaccard Distance** is a statistic that measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and obtained by subtracting Jaccard Index from unity: d(A,B) = 1- J(A,B).
* **Triangle Inequality Property:** It states that for any triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side, d(a,c) <= d(a,b) + d(b,c).

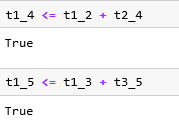
All pairwise Jaccard Distance among 6 documents is been calculated and snippet is shown below:



Property of Triangle Inequality is tested against few examples:

In the snippet given below: tx-y is the Jaccard distance between document x and y. Example: t1-4

Triangle Inequality measure holds true for Jaccard distance every time.

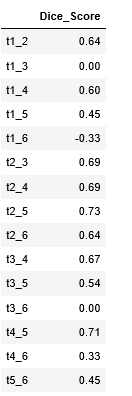


1. **Now implement the difference function for the Dice Coefficient and show that the property of triangle inequality may not hold for this measure.**

**Ans1b.**

* **Dice Coefficient:** It is a statistic used to measure the similarity of two samples, defined as twice the size of the intersection divided by the size of the union of the sample sets: DSC(A,B) = 2\*|A & B|/|A|B|.
* **Dice Distance:** It is defined as semi metric version of Jaccard Index(not a proper distance metric) because it does not hold true for the triangle Inequality measure every time, defined as subtracting Jaccard Index from unity: d(A,B) = 1- DSC(A,B).

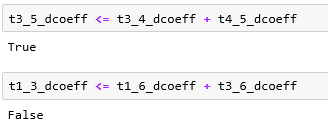
All pairwise Dice Distance among 6 documents is been calculated and snippet is shown below in which t1\_2 denotes the distance between document1 and 2. Similarly rest are calculated.



Property of Triangle Inequality is tested against few examples in this case as well the snippet given below.

The tx-y in the snippet shows the Dice Coefficient distance between document x and y. Example: t1-4

Triangle Inequality measure may not hold true every time for Dice Coefficient measure of similarity.

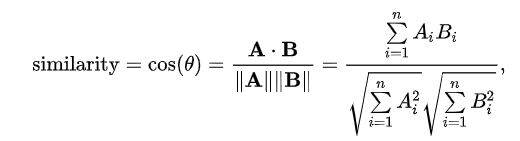


==================================================================================

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

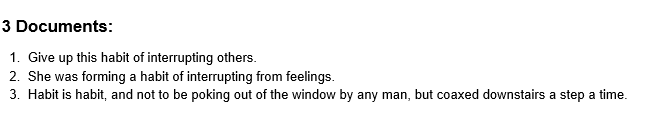
**Ans2.**

**Cosine Similarity:** It is a metric used to measure how similar the documents are irrespective of their size. It is based on count of the maximum number of common words between the documents. The smaller the cosine, more similar the vectors.

It can be calculated as: 

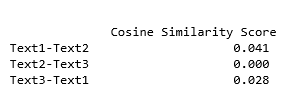
1. **Find 3 short documents about which you might want to know their similarity. Produce 5 variants on one of the documents and see how the cosine similarity changes.**

**Ans2a.** 3 similar short documents are taken into consideration.

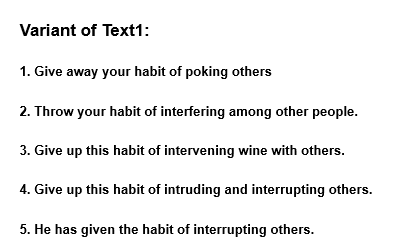


Cosine Similarity between the 3 documents is calculated using the get\_cosine() from the cosine.py file which internally calculates the Vector dictionary containing the words and their respective tf-idf values and shown below:

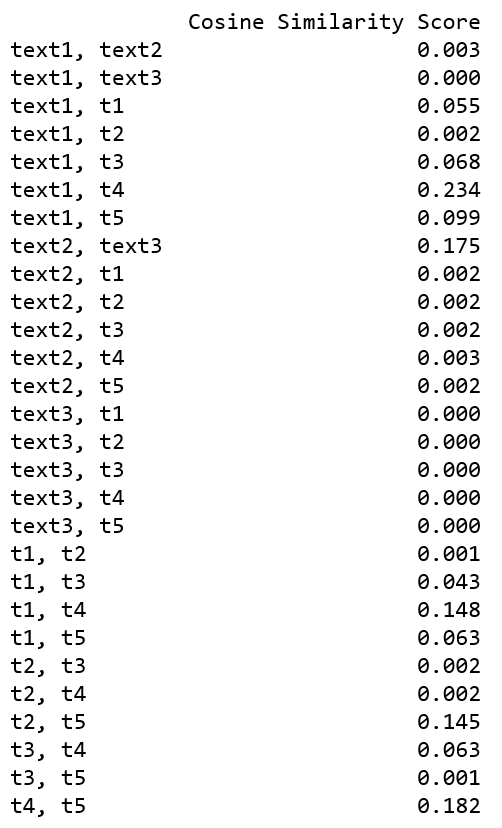
The document 1 and 2 are having a little similarity, so their cosine similarity is coming as a higher value whereas document 2 and 3 are completely dissimilar. Thus their cosine similarity is showing as 0.



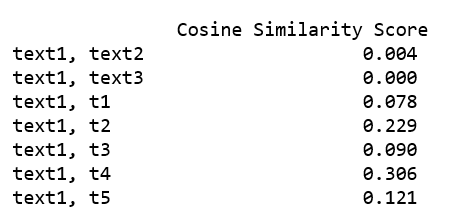
5 variants of the document 1 are generated and cosine similarity among all 8 documents (3 original + 5 variants) is calculated and the snippet is shown below:



The cosine similarity of the document 1 with its variant is coming higher compared to the other 2 documents as the variants are more like the document 1 rather than document 2 and 3.



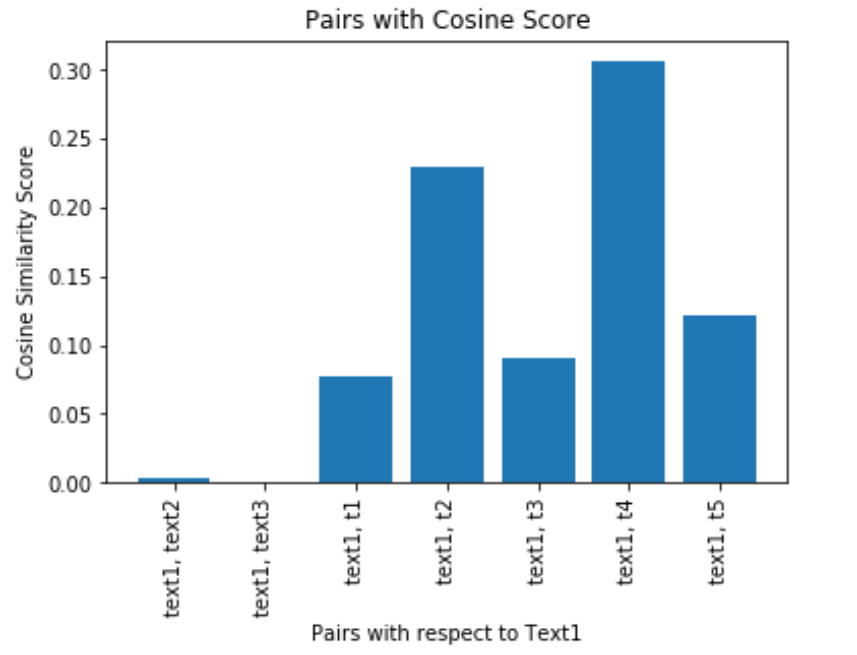
Cosine Similarity of document 1 against other documents and its variants is shown below which also shows higher cosine similarity among the variants and their base :



1. **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.**

**Ans2b.** Cosine similarity score of Document 1 against all other documents and its variants is shown below which clearly proves higher the cosine similarity score, higher is the similarity among the documents.

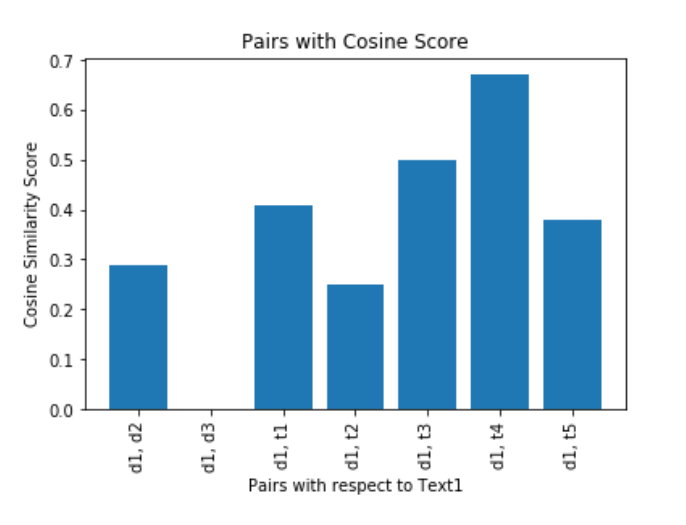
The cosine angle between the documents is low, it implies that the more similar the documents are to each other. Hence intuitions about differing documents less similar lead to scores that are less similar are apt and appropriate.



1. **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?**

**Ans2c.** Python package to compute cosine similarity and euclidean distance is ‘sklearn.metrics.pairwise.cosine\_similarity’ and ‘sklearn.metrics.pairwise.euclidean\_distances’.

Cosine similarity of document 1 against all other documents and its variant is calculated using the library which internally generates the Term frequencies for the most occurring terms using CountVectorizer’s fit\_transform and then using the cosine\_similarity function. The cosine similarity generated manually and through library are co relating to each other whereas the values are differing due to the tf idf values calculated manually and through CountVectorizer. The plot is shown below:

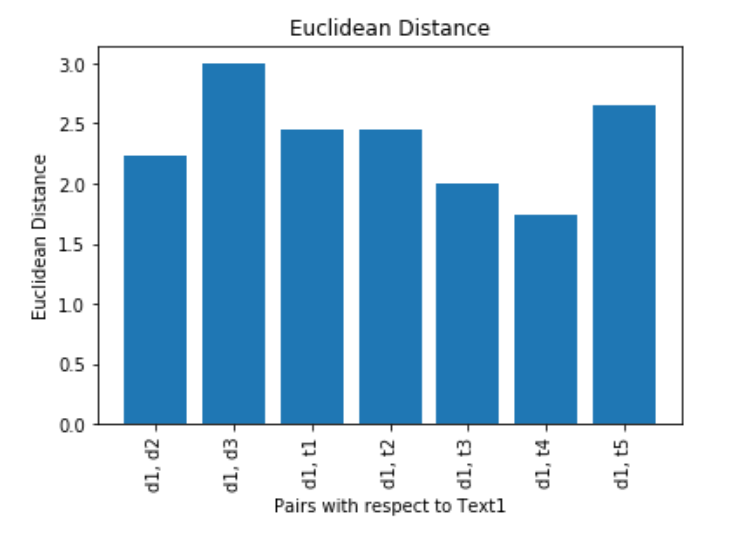


Further the Euclidean distance between the document 1 and other documents and its variants are calculated using the euclidean\_distances() of sklearn.metrics. The plot is shown below:

Euclidean distance plot is opposite to the cosine similarity plot.

Euclidean distance measures the length between two sample points rather than similarity. It considers the magnitude of the vector whereas Cosine similarity considers the angle between the vectors.

Cosine similarity is generally used for measuring distance when the magnitude of the vectors does not matter that is the word count. [1]



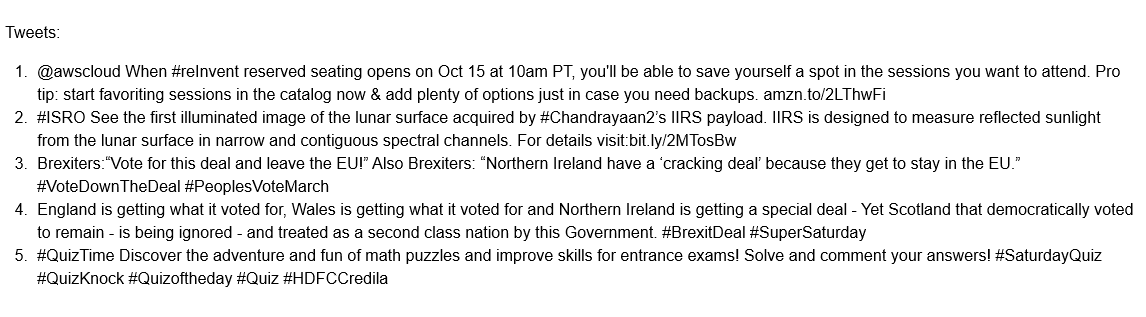
==================================================================================

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

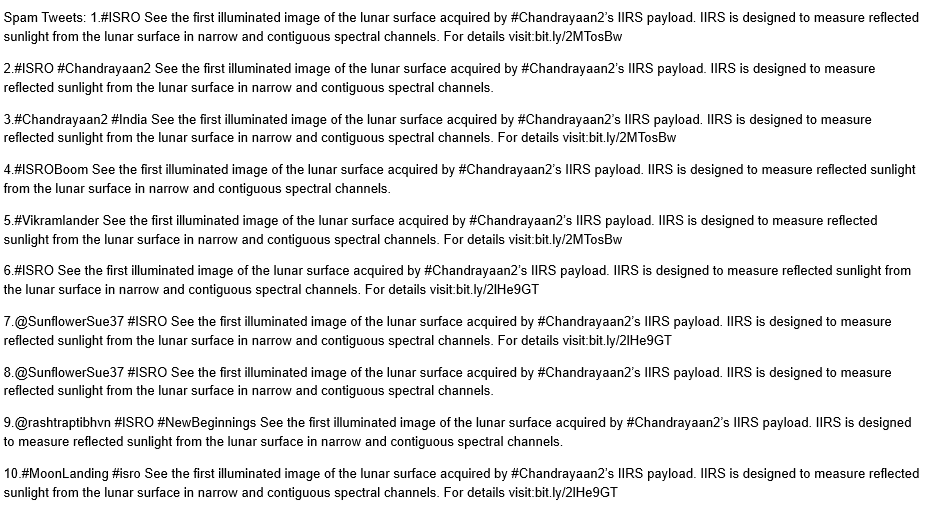
**Now, perform comparisons between these 20 SPAM tweets each of the 5 Normal Tweets. Plot their edit-distance scores in a graph and colour code to show how the SPAM v Normal ones. Are the SPAM tweets obvious, if not why?**

**Ans3.**

Snippet of 5 normal tweets is shown below:



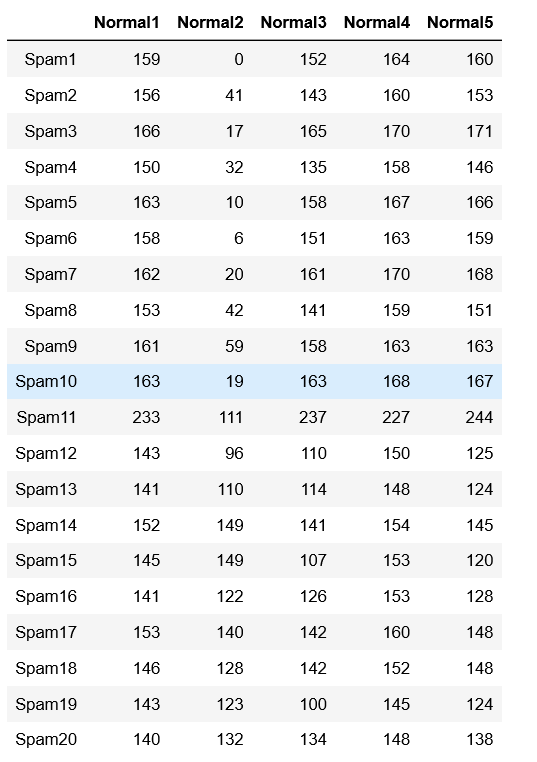
Snippet of 20 spam tweets generated from tweet number 2 is shown below:



All the normal tweets and the spam tweets are pre processed which consists of the steps(Tokenisation, Normalization(Lowercase, stopwords removal) followed by punctuation marks removal) with the help of Tweet Tokenizer.

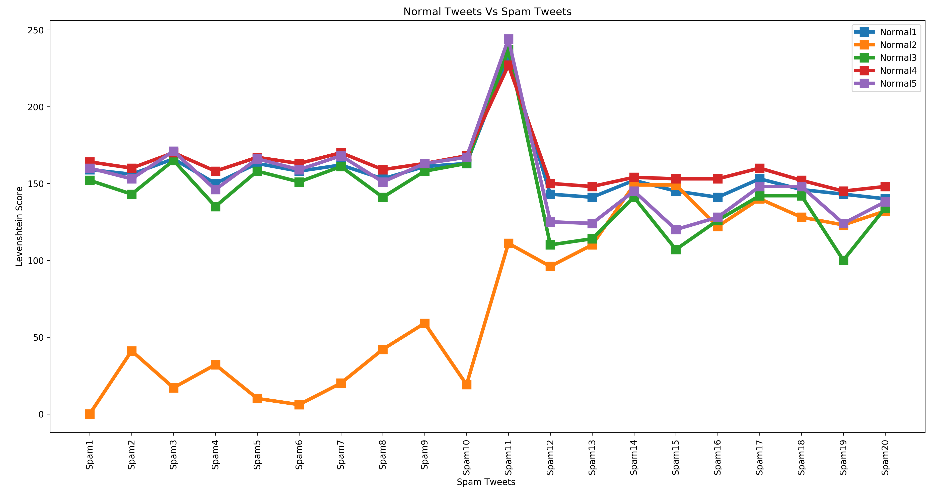
* **Levenshtein Distance**: It is a string metric for measuring the difference between two sequences.

Levenshtein distance of all spam tweets with respect to normal tweets is calculated and snippet is shown below:



A plot of edit-distance score is shown below.

The graph depicts that the distance between spam tweets and the normal tweets. Spam tweets are obvious due to similarity between them and normal tweet whose variants they are as distances between them is less as compared to that of spam tweets and other normal tweets.



=========================================================================================

**REFERENCES**

[1] (Chris Emmery, 2019, https://cmry.github.io/notes/euclidean-v-cosine)