Question 1 Pick a category of things and describe 5 instances of this category using 5 feature-words. Re-use some of the feature-words across different instances so that similarities between instances can be found.

Category chosen for analysis is : **Skills of game for ranked tennis players**

[1] Federer : [‘reach’, ’forehand’, ‘accuracy’, ‘variety’, ‘volleys’]
[2] Nadal : [‘power’, ’forehand’, ‘stamina’, ‘mentality’, ‘volleys’]
[3] Djokovic : [‘mentality’, ’backhand’, ‘stamina’, ‘mentality’, ‘reach’]
[4] Thiem : [‘power’, ’backhand’, ‘stamina’, ‘projection’, ‘winners’]
[5] Zverev : [‘power’, ‘crosscourt’, ‘reach’, ‘forehand’, ‘stamina’]

1(a) Modify the Jaccard-Index program given in the lecture to do Jaccard-Distance and find the Jaccard-Distance between all the pairwise-comparisons of the 5 instances. Show the distance-values found in a matrix with rows and columns for the 5 instances.

Jaccard distance between two sample sets is measure of dissimilarity between sample sets. It is obtained by subtracting the Jaccard Index or Jaccard’s Coefficient from 1. Jaccard coefficient is ratio of length of intersection of sets to length of union of sets.

$$Coeff_{Jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$d_{Jaccard}(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

Python function can be implemented for calculating Jaccard distance of 2 sets.

```
def jaccardDistance(instance1, instance2):
    set_1 = set(instance1)
    set_2 = set(instance2)
    ans = float(len(set_1 | set_2) - (len(set_1 & set_2))) / len(set_1 | set_2)
    return round(ans, 2)
```

Operation of Jaccard distance calculation between Federer and Nadal is explained below:

- (Federer ∪ Nadal) = [Combine skills of Federer and Nadal → then reduce duplicates]
  =[‘reach’, ’forehand’, ‘accuracy’, ‘variety’, ‘volleys’, ‘power’, ‘stamina’, ‘mentality’]
- (Federer ∩ Nadal) = [Keep skills common to both Federer and Nadal]
  =[‘volley’, ‘forehand’]

Hence, $d_{Jaccard}(F,N) = \frac{N(Federer \cup Nadal) - N(Federer \cap Nadal)}{N(Federer \cup Nadal)} = \frac{8-2}{8} = 0.75$

Similar Calculations give us **matrix of Jaccard Distances** between all instances. Due to symmetry of distance metric; matrix is symmetric about its diagonal. Diagonal is naturally 0, since numerator term i.e. difference between union and intersection of set with itself is 0.

Higher the dissimilarity, higher the Jaccard Distance. We can gauge dissimilarity between feature sets using Jaccard Distance. As game of Federer and Thiem is different, they share no quality; hence distance is Maximum. Whereas Nadal and Zverev both have good stamina, powerful shots and nice forehand; hence their distance is minimized.

	Federer	Nadal	Djokovic	Thiem	Zverev
Federer	0	0.75	0.89	1	0.75
Nadal	0.75	0	0.75	0.75	0.57
Djokovic	0.89	0.75	0	0.75	0.57
Thiem	1	0.75	0.75	0	0.75
Zverev	0.75	0.57	0.57	0.75	0

Table 1 Jaccard Distance matrix

1(b) Check whether property of the triangle inequality holds for the Jaccard-Distance measure on the basis of the values found. Show the results of the comparisons made in testing for this property.

Triangle inequality states that if the distance between x and y is small and the distance between y and z is small, then the distance between x and z cannot be very large. This can be simply formulated using geometry of Triangle: Sum of 2 sides of triangle is always greater than third side.

```
if (ab + bc > ac) and (bc + ac > ab) and (ab + ac > bc)
```

Adjoined python script is used to verify Triangle inequality.

```
def triangle_ineq_Jaccards(features):
    a, b, c = features[0], features[1], features[2]
    ab = jaccardDistance(a, b)
    bc = jaccardDistance(b, c)
    ac = jaccardDistance(a, c)
    if (ab + bc > ac) and (bc + ac > ab) and (ab + ac > bc):
        print('ab->', ab, 'bc->', bc, 'ac->', ac)
        print("Holds Triangle Inequality")
        return True
    else:
        print('ab->', ab, 'bc->', bc, 'ac->', ac)
        print("Doesnt Hold")
        return False
```

Figure 1 Triangle inequality for Jaccard's distance

Upon running script, I could verify that, **Triangle inequality holds for Jaccard's Distance**. For example:

```
Checking for thiem,nadal and djokovic
ab-> 0.75 bc-> 0.75 ac-> 0.75
Holds d(Jaccards) Triangle Inequality
Checking for thiem,nadal and zverev
ab-> 0.75 bc-> 0.57 ac-> 0.75
Holds d(Jaccards) Triangle Inequality
Checking for thiem,djokovic and federer
ab-> 0.75 bc-> 0.89 ac-> 1.0
Holds d(Jaccards) Triangle Inequality
Checking for thiem,djokovic and nadal
ab-> 0.75 bc-> 0.75 ac-> 0.75
Holds d(Jaccards) Triangle Inequality
```

Figure 2 Triangle inequality holds for Jaccard's Distance

1 (c) Implement the Dice Coefficient and compare its results to those found for Jaccard Distance; comment on the what seems to be the similarity and differences in the results from these two measures.

$$Coeff_{dice}(A,B) = \frac{2|A \cap B|}{|A| + |B|}$$

$$d_{dice}(A,B) = \frac{|A| + |B| - 2|A \cap B|}{|A| + |B|}$$

Dice Coefficient or **Sørensen - Dice Coefficient** compares similarity of sample sets. As against Jaccard's Index, Dice coefficient doubles the numerator and does not eliminate set intersection from denominator.

```
Checking for federer,djokovic and thiem
ab-> 0.2 bc-> 0.4 ac-> 0.0
Doesnt Hold
Checking for federer,djokovic and zverev
ab-> 0.2 bc-> 0.6 ac-> 0.4
Holds Dice coeff Triangle Inequality
Checking for federer,thiem and nadal
ab-> 0.0 bc-> 0.4 ac-> 0.4
Doesnt Hold
Checking for federer,thiem and djokovic
ab-> 0.0 bc-> 0.4 ac-> 0.2
Doesnt Hold
```

Figure 3 Triangle inequality Does not hold for Dice Coefficient

Dice coefficient Vs Jaccard's distance

Dice coefficient is measure of SIMILARITY as against Jaccard's Distance which expresses DISSIMILARITY. As observed in table 2, similarity of instances with themselves is maximized [Djokovic - Djokovic → 1]. Triangle inequality DOES NOT hold true for all comparisons when Dice coefficient is treated as a distance metric. For example, Federer and Thiem have no shared skills; but both Federer and Thiem share game qualities with Djokovic. But treating Djokovic as common "Base of triangle" for comparison, we cannot establish a similarity between Federer and Thiem.

```python
def dice_coefficient(instance1, instance2):
    """dice coefficient = 2(overlap of sets)/(len(set1) + len(set2))."""
    set_1 = set(instance1)
    set_2 = set(instance2)
    return (len(set_1 & set_2) * 2.0) / (len(set_1) + len(set_1))
```

	Federer	Nadal	Djokovic	Thiem	Zverev
Federer	1	0.4	0.2	0	0.4
Nadal	0.4	1	0.4	0.4	0.6
Djokovic	0.2	0.4	1	0.4	0.6
Thiem	0	0.4	0.4	1	0.4
Zverev	0.4	0.6	0.6	0.4	1

Table 2 Dice Coefficient matrix

Dice distance (which is calculated by subtracting Dice coefficient from 1) is a comparable measure against Jaccard's distance. It is evident from Table 1 and 3 that differences are boosted and similarities are punished. Unlike Jaccard's distance, Dice Distance does not follow Triangle inequality. However, **I observed for my random dataset that:**

- Jaccard's Index as well as Dice coefficient DOES NOT follow Triangle inequality.
- Whereas, Jaccard's distance and Dice distance DO follow triangle inequality.

```python
def dice_distance(instance1, instance2):
    """dice distance = 1 - [2(overlap of sets)/(len(set1) + len(set2))]"""
    set_1 = set(instance1)
    set_2 = set(instance2)
    return 1 - ((len(set_1 & set_2) * 2.0) / (len(set_1) + len(set_1)))
```

	Federer	Nadal	Djokovic	Thiem	Zverev
Federer	0	0.6	0.8	1	0.6
Nadal	0.6	0	0.6	0.6	0.4
Djokovic	0.8	0.6	0	0.6	0.4
Thiem	1	0.6	0.6	0	0.6
Zverev	0.6	0.4	0.4	0.6	0

Table 3 Dice Distance matrix

Question 2 Refer the Cosine.py program provided [Install the necessary packages]. Identify 3 online articles about the same event (e.g., Trump getting COVID-19*) and 3 online articles about a related, but different, event (e.g., Johnson getting COVID 19*). Select some key paragraphs from the articles an run them through your preprocessing pipeline (use aggressive stop-word removal).

SET 1: Article about event: Ireland accepting Vaccine when available

[1] **d1:** more than half of irish people would take a covid 19 vaccine if one became available, according to a new survey carried out on behalf of the irish pharmaceutical healthcare association. three fifths of men said they would take the vaccine, while half of women said they would take vaccine. younger people were the least likely to take the vaccine and nearly one - fifth are saying they would not take vaccine.

[2] **d2:** almost a third of irish people would be unlikely to take the first publicly available eu-approved covid vaccine. on the other hand, 56 per cent of people say they're likely take the vaccine and twelve per cent don't know. but, as presenter mark coughlan pointed out, experts say 75 per cent of the population will need to get the vaccine to develop herd immunity.

[3] **d3:** just over half of irish citizens are willing to take a coronavirus vaccine, an irish pharmaceutical healthcare association survey reveals. females are less willing to take a covid 19 jab once it becomes available. one-third were unsure about accepting the injection, while twelve per cent said they would not do so. men and the elderly are interested in taking the vaccine. just under 1,000 people were quizzed over telephone poll at the start of october.

SET 2: Article about event: India accepting Vaccine when available

[1] **d4:** over 60 per cent of indians are skeptical about a vaccine for covid 19 even if it becomes available in 2021, according to a recent survey by social media engagement platform localcircles. just 12 per cent of respondents said they were willing to get vaccinated and return to pre covid 19 lifestyles. the country is the second-worst affected in the world with nearly eight million cases.

[2] **d5:** the survey was conducted to know the tentative perception of reaction from people when the government takes covid 19 vaccine to its people. it also sought to know current behavior of people facing covid 19 threats, and how long will they continue to endure the suffering from the pandemic. the survey received over 25,000 responses from over 225 districts of india.

[3] **d6:** scientists from around the world, including india are developing several vaccines that are at different stages of clinical trials. only 12 per cent respondents said would get vaccinated and go back to living pre covid lifestyle, while 25 per cent said they will get vaccinated but still will not go back to pre covid lifestyle, and 10 per cent said they will not take it at all in 2021.

I did run the inputs through pre-processing pipeline. Additionally, punctuations and empty strings were removed using python string library. For example, After preprocessing and TF-IDF indexing; following vector dictionaries are obtained. Each document dictionary is sorted for documentation.

```
# remove punctuation from each word - reference !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
table = str.maketrans('', '', string.punctuation)
words = [w.translate(table) for w in words]
# remove singleton empty strings after punctuation removal.
words = filter(None, words)
```

Figure 4 Punctuation and empty-string removal

d1 {'one': 1.5563, 'half': 0.9542, 'take': 0.8805, 'women': 0.7782, 'carried': 0.7782, 'least': 0.7782, 'fifth': 0.7782, 'new': 0.7782, ,……, 'nearly': 0.4771, 'available': 0.1761, '19': 0.1761, 'survey': 0.1761, 'covid': 0.0}
d2 {'say': 1.5563, 'mark': 0.7782, 'approved': 0.7782, 'need': 0.7782, 'hand': 0.7782, 'population': 0.7782, 'they're': 0.7782, 'don't': 0.7782, ……..'vaccine': 0.2375, 'would': 0.1761, 'available': 0.1761, 'covid': 0.0}
d3 {'willing': 0.9542, 'citizens': 0.7782, 'jab': 0.7782, 'females': 0.7782, 'interested': 0.7782, 'taking': 0.7782, 'reveals': 0.7782, 'onethird': 0.7782, ….. 'per': 0.1761, 'cent': 0.1761, 'vaccine': 0.1584, 'covid': 0.0}
d4{'engagement': 0.7782, 'lifestyles': 0.7782, 'country': 0.7782, 'indians': 0.7782, 'affected': 0.7782, 'recent': 0.7782, 'media': 0.7782, …. 'said': 0.1761, 'survey': 0.1761, 'vaccine': 0.0792, 'covid': 0.0}
d5{'know': 0.9542, 'sought': 0.7782, 'endure': 0.7782, 'government': 0.7782, 'continue': 0.7782, 'facing': 0.7782, 'behavior': 0.7782, …..'india': 0.4771, '19': 0.3522, 'survey': 0.3522, 'vaccine': 0.0792, 'covid': 0.0}
d6{'back': 1.5563, 'lifestyle': 1.5563, 'go': 1.5563, 'pre': 0.9542, 'vaccinated': 0.9542, 'vaccines': 0.7782, 'developing': 0.7782, 'trials': 0.7782, ……'world': 0.4771, 'india': 0.4771, 'respondents': 0.4771, '2021': 0.4771, 'would': 0.1761, 'take': 0.1761, 'covid': 0.0}

2(a) Taking the pre-processed files, find the pairwise cosine-similarity for articles within each group (the Trump and the Johnson groups) and then between the articles in both groups (i.e., comparing the Trump ones and the Johnson ones).

Vector space modelling is tool to find similar documents. We have obtained couple of sets of documents. During pre-processing, English stop words and punctuations are removed. Then we obtain TF-IDF indices for key words – Larger index indicates rarity of term across documents. TFIDF has higher weight when word has high TF (within document d) and a low DF (within entire corpus D).

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. Independence from document size is distinct feature of Cosine similarity. I used provided get_cosine() function which is SET oriented implementation to calculate cosine similarity between all documents**.**

	d1	d2	d3	d4	d5	d6
d1	1	0.442627	0.53886	0.295312	0.313527	0.146059
d2	0.442627	1	0.319438	0.261426	0.265306	0.222234
d3	0.53886	0.319438	1	0.340997	0.255551	0.168655
d4	0.295312	0.261426	0.340997	1	0.239641	0.452898
d5	0.313527	0.265306	0.255551	0.239641	1	0.121218
d6	0.146059	0.222234	0.168655	0.452898	0.121218	1

Table 4 Cosine similarity between documents

Naturally, self-distance for all documents is highest i.e. 1

Ideally, cosine similarity metric for documents within a set should be maximized compared to documents from other set. I have color coded the chart where –

When metric is greater for "InSet" document, it is indicated with **YELLOW**

When metric is smaller than "OutSet" document, it is indicated with **GRAY**

Observations:

- **SET 1 [d1,d2,d3]:** For Document d1 and d2; similarity within Set 1 is greater rather than with set 2. However, d3 is more similar with d4, rather than d2.
- **SET 2 [d4,d5,d6]:** Except for correlation between d4 and d6, all documents have higher similarity with documents from set 1.
- **D5 is related to covid vaccine, but it talks little about actual proportion of pupil accepting or rejecting vaccination. Hence, it has very little similarity metric across all documents.**

3(b) Plot these similarity scores in a graph and check if the scores cluster to reflect the two groups. Comment on the clustering/lack-of clustering (whichever happens) and trace the sources of these effects to the feature-words from each group.

Table 4 is materialized graphically in adjoined figure 5. Ideally, similarity within set must be high, hence forming a cluster, i.e. cosine index between d1-d2-d3 should be high as against between d1 and d4-d5-d6 should be very low etc. But **Clustering within set is NOT observed for taken documents.**

Instead, **cross-set cluster between d2-d3-d4 is seen.** Also, a sporadic result of similarity is seen between d2 and d5.

Figure 5 Cosine Similarity graphs for set1, set2 and entire corpus

Inference:

- VSM and cosine distance metric works strictly on numbers underlying. Lemma behind the TFIDF score is not considered while calculating metric.
- Similarity between d1 and d3 is governed by sequences like "half of irish citizens", "pharmaceutical healthcare association". Also, d4 and d6 correlate based on congruent sequences "12 per cent", "pre covid lifestyle".

3(c) Find the sklearn python package that allows you to compute Cosine Similarity and, also, Manhattan Distance. Use it to process the data you have already. Are the scores you find from the sklearn method the same as the ones from the Cosine.py program? Do the Manhattan Distance scores for the same item-comparisons change markedly from those found for Cosine Similarity; discuss what you find.

Scikit-learn functionality cosine_similarity() and Manhattan_distances() are utilized to cross verify earlier results with opensource library. Count_vectorizer is used to format preprocessed TFIDF dictionaries [refer Question2] into pandas data frames. I utilized matplotlib function to plot values for cosine similarities between all documents.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity, manhattan_distances
```

Figure 6 shows comparison of manual and sklearn based cosine similarity. MY understanding from observation of graphs is:

- Similarity between documents show similar resemblance using sklearn library. **Trend of similarities is mostly preserved.**
- Sklearn results are more pronounced and at times, similarity value is 2-4 folds than earlier.
- **Noticable difference is related to d5.** Cosine scores for d5 were punished in manual calculation of cosine similarity since it did not precisely provide percent of people supporting or opposing vaccination like other documents. But, manual calculations did not capture relevance of d5 to the underlying topic of all documents : "Perception of citizens towards covid 19 vaccine". It is noticed by sklearn version of solution.

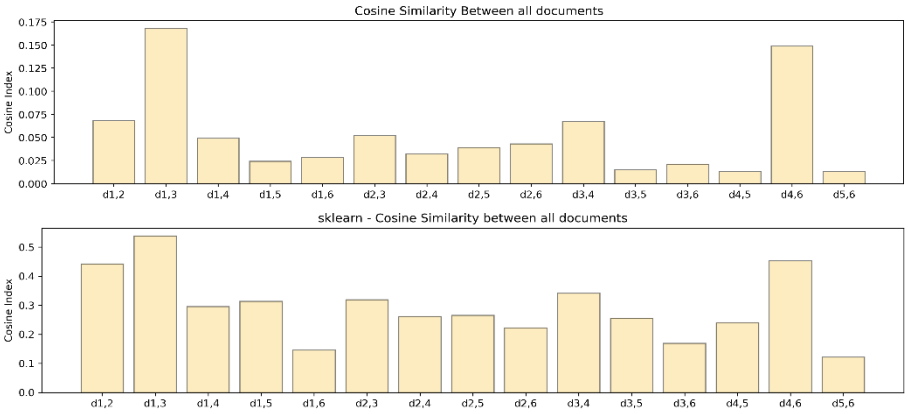
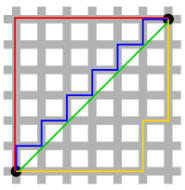


Figure 6 Cosine indices - manual Vs sklearn

Manhattan Distance is unit cartesian distance between 2 vector space features. It is vector oriented L1 norm metric for dissimilarity. In adjoined figure, green line is Euclidean distance, whereas remainder are Manhattan distances. For analogy, consider **p and q → d_x and d_y**. So, cumulative distance between TFIDF vectors for document pairs is calculated to get Manhattan distance. Manhattan distance is said to outperform cosine metric during LSTM implementation with both better precision and recall[Link]. **Since Manhattan is a distance metric, naturally it has inverse relation with similarity metric [distance = 1 - similarity]. This is shown in figure 7.**

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$



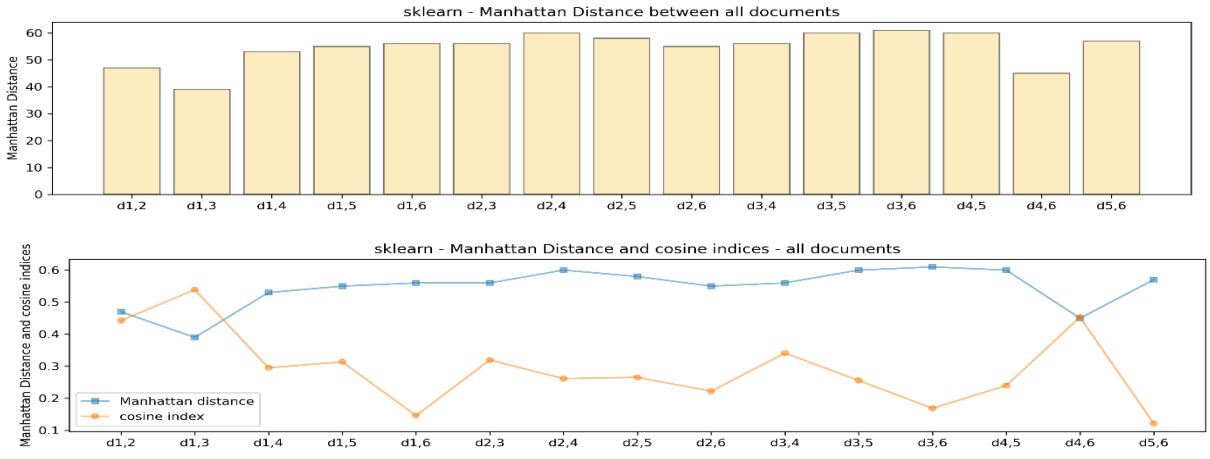


Figure 7 Inverse relation between Cosine and Manhattan distance