# IT458 Assignment 2

NAME: SUYASH CHINTAWAR
ROLL NO.: 191IT109
TOPIC: VSM and TF-IDF

Note:
1) The colab link has been attached below. After opening the link, if it opens in drive, click on “Open with Google Colaboratory” to view the complete code.

**Colab notebook link:**
https://colab.research.google.com/drive/1wVLhnzhyiZT4-DgffFosRPzQyNw27YNo

**Q. Demonstrate the process of generating term weights of the vocabulary as per the standard TF-IDf weighting scheme.**

We use the same articles extracted from the Incredible India website. The same preprocessing has been considered for the TF-IDF index terms as well.

```
Tokenized texts:
 [['Standing', 'on', 'the', 'waterfront', 'of', 'Lake', 'Pichola', 'at', 'Gangori', 'Ghat', 'is', 'the', 'stunning', 'Bagore', 'ki', 'Haveli.', 'Constructed', 'in',
Number of tokens after tokenization: 3680

Normalized texts:
 [['standing', 'on', 'the', 'waterfront', 'of', 'lake', 'pichola', 'at', 'gangori', 'ghat', 'is', 'the', 'stunning', 'bagore', 'ki', 'haveli', 'constructed', 'in', '
Number of tokens after normalization: 2969

Lemmatized texts:
 [['standing', 'on', 'the', 'waterfront', 'of', 'lake', 'pichola', 'at', 'gangori', 'ghat', 'is', 'the', 'stunning', 'bagore', 'ki', 'haveli', 'constructed', 'in', '
Number of tokens after lemmatization: 2749

Texts after stopword removal:
 [['standing', 'waterfront', 'lake', 'pichola', 'gangori', 'ghat', 'stunning', 'bagore', 'ki', 'haveli', 'constructed', '18th', 'century', 'amar', 'chand', 'badwa',
Number of tokens after lemmatization: 2652
```

Thus, we can see that we have a total of 2652 index terms across all 100 documents scraped.

```
# Set of all index terms across all documents
tokens = get_all_tokens(preprocessed_texts,False)
print('Total no. of index terms:',len(tokens))
```

```
Total no. of index terms: 2652
```

Step 1: Term frequencies computation for each (term,document) pair.

```
# Computing term frequency using two methods, log normzalization and double normalization-K
def tf(term, document,tf_type='logNormal',k=0.3):
  '''
  This function computes term freq for a index term in a document
  '''
  freq = document.count(term)
  if tf_type=='logNormal' and freq > 0:
    tf = np.log2(freq) + 1
    return tf
  # elif freq > 0 and tf_type=='doubleNormalK':
  elif tf_type=='doubleNormalK':
    max_count = document.count(max(set(document), key=document.count))
    tf = k + (1-k)*(freq/max_count)
    return tf
  else:
    return 0
```

The function above is used to calculate the term frequency with an option to choose any one of the two variants namely, Log Normalization and Double Normalization-K. The value of 'K' was set to 0.3. It can be changed on requirement by altering the parameter values. By default, the Log Normalization variant is set to compute term frequency if not specified. This method computes the TF for a single term.

Step 2: Calculating the IDF for each term across all documents.

```
def idf(term,documents,idf_type='invFreq'):
  '''
  This function computes inverse doc freq for a index term for all documents
  Three methods used,
  1) inverse frequency -> default
  2) inverse frequency smooth
  3) probabilistic inverse frequency
  '''
  N = len(documents)
  ni = 0
  for document in documents:
    if term in document:
      ni += 1
  if idf_type=='invFreq':
    return np.log2(N/ni)
  elif idf_type=='invFreqSmooth':
    return np.log2(1 + (N/ni))
  elif idf_type=='invFreqProb':
    return np.log2((N-ni)/ni)
```

Three variants of IDF have been implemented namely,

- Inverse frequency
- Inverse Frequency Smooth
- Probabilistic inverse frequency

This method computes the IDF for a single term.

Step 3: Generating term weights for each (term,document) pair.

```
[22] def get_term_weights(documents, tokens, tf_type='logNormal', idf_type='invFreq'):
       '''
       Computes term weights (TF-IDF) for each (index term, document) pair
       Returns a matrix of dimension (m,n), where
       'm'-> no. of docs
       'n'-> no. of index terms
       '''
       term_weights = get_tf(documents,tokens,tf_type)
       term_weights = np.array(term_weights,dtype=float)
       idf = get_idf(documents,tokens,idf_type)
       idf=np.array(idf,dtype=float)
       for item in idf:
         if item<0:
           print(item)

       for i in range(len(term_weights)):
         term_weights[i] = np.multiply(term_weights[i],idf)

       return term_weights
```

The ‘get_tf’ method computes the term frequency matrix for each term-doc pair and the ‘get_idf’ method computes the IDF of all index terms across the documents. So if we have 100 docs and 2652 index terms, ‘get_tf’ will return a (100, 2652) dimensional matrix whereas ‘get_idf’ will return a (1, 2652) dimensional vector. Each of the document vector is multiplied by the IDF vector using 1:1 vector multiplication (Hadamard product). This gives us the final term weights for the index terms in all documents yielding a matrix of dimension (100, 2652).

**Q. Represent the document corpus using the standard TF-IDF weights as per the formalism used by the vector space IR model. Demonstrate the process of generating the ranked list for sample queries using any one distance measure and a similarity measure. Compare and contrast the rankings generated by each.**

The standard TF-IDF weighting scheme is taken when the TF variant used is Log Normalization and IDF variant is set to Inverse Frequency. From the previous question, we represented the document corpus using the standard TF-IDF weighting scheme and obtained the term weight matrix.

| | | sambhar | ground | staple | construction | heart | businessmen | include | governance | afghan | ... | sword | birdlovers | surrounding | comprising | tall | thereby | doused | upper |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 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 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 3.643856 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **95** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 5.643856 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **96** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **97** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **98** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **99** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 6.643856 | ... | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

100 rows × 2652 columns

The above dataframe is created to visualize the term weights. The columns represent the index terms and rows represent document IDs. The matrix will be sparse as not all documents contain many index terms. But some non zero values can be seen in the above matrix as well.

The next step is to generate the ranked lists for a given query. We use Euclidean distance as the distance measure and Cosine similarity as the similarity measure. The methods for computing the same are depicted below,

```
# Euclidean distance between two numpy arrays
def euclidean_dist(a,b):
  return  np.linalg.norm(a-b)

# Cosine similarity between two numpy arrays
def cosine_sim(a,b):
  return  1 - spatial.distance.cosine(a,b)
```

To generate the ranking list for a given query the steps are as follows,

- First step is used to represent the query in the same vector space as the document corpus. The process shown below is similar to the term weighting process done for the document corpus.

```
# represent query in same vector space
tf_query = []
for token in tokens:
  tf_query.append(tf(token,query,tf_type))

term_weight_query = np.array(tf_query,dtype=float)
idf = get_idf(documents,tokens,idf_type)
idf=np.array(idf,dtype=float)
term_weight_query = np.multiply(term_weight_query,idf)
```

- The next step is to compare the query vector with each of the document vectors using the specified distance/similarity measure, either euclidean or cosine.

```
# compare with distance measure
if measure=='euclidean':
  for i in range(len(term_weights)):
    distance = euclidean_dist(term_weight_query,term_weights[i])
    ranked_list.append([distance,i])

  ranked_list.sort()
  return ranked_list

elif measure=='cosine':
  for i in range(len(term_weights)):
    similarity = cosine_sim(term_weight_query,term_weights[i])
    ranked_list.append([similarity,i])

  ranked_list.sort(reverse=True)
  return ranked_list
```

- Once the distances with each of the documents have been computed the distances are sorted to get the final ranking lists of the documents.

Three sample queries have been considered each of them consisting of 2,3 and 5 words respectively, the sample queries are as follows,
Query 1 : ['deepfried','flatbread']
Query 2: ['standing','waterfront','pichola']
Query 3: ['traditional','curry','pakoda','boiled','rice']

The ranked lists are then generated and the top 10 ranked documents are shown. When euclidean distance is used as a measure, the outputs for each of the query is shown below,

Query 1:

```
*********************************
QUERY NO: 0
QUERY: ['deepfried', 'flatbread']

TOP 10 RANKINGS:

Document ID     Euclidean Distance from query
92               16.598756322494378
94               18.693466571080926
30               19.050018404672496
21               19.955239333959053
96               20.21986732106013
64               20.78139648459215
44               21.285727113878064
23               21.898325659033606
91               22.811620305237366
14               24.85070060976365

TOP 3 RANKED DOCUMENTS:
Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 94
Content:
 located ajmer road small village located35 km jaipur known traditional bagru print

Document ID: 30
Content:
 kadhi traditional curdbased curry fried gram flour fritter called pakoda often eaten boiled rice roti indian flatbread

*********************************
```

Query 2:

```
*********************************
QUERY NO: 1
QUERY: ['standing', 'waterfront', 'pichola']

TOP 10 RANKINGS:

Document ID     Euclidean Distance from query
94               20.130787351066463
92               20.225276387755393
30               21.101219965007175
21               21.307615629284378
96               21.555647552579178
64               22.083236782219146
44               23.665795917537046
91               24.55052552886749
14               25.949185492532227
23               26.283453858928997

TOP 3 RANKED DOCUMENTS:
Document ID: 94
Content:
 located ajmer road small village located35 km jaipur known traditional bagru print

Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 30
Content:
 kadhi traditional curdbased curry fried gram flour fritter called pakoda often eaten boiled rice roti indian flatbread

*********************************
```

Query 3:

```
*******************************
QUERY NO: 2
QUERY: ['traditional', 'curry', 'pakoda', 'boiled', 'rice']

TOP 10 RANKINGS:

Document ID     Euclidean Distance from query
30               15.20703652904809
92               18.947482120591044
94               20.277979725567395
96               21.25910017086301
21               21.878665575940694
64               22.634727088217453
44               24.181220399083283
91               25.04775094339016
14               26.42010127354498
23               26.748485383904974

TOP 3 RANKED DOCUMENTS:
Document ID: 30
Content:
 kadhi traditional curdbased curry fried gram flour fritter called pakoda often eaten boiled rice roti indian flatbread

Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 94
Content:
 located ajmer road small village located35 km jaipur known traditional bagru print

*******************************
```

Observations: We can see that shorter documents are always favored when euclidean distance measure is considered because the query is generally small. So, we can see that in query 2 (with three words), the top ranked documents are nowhere relevant to the query. It can also be seen that the same documents are ranked higher in all the queries because of the small size of the query. This behavior of euclidean distance measure can be curbed by normalizing the document and query vectors.

When cosine similarity is used as a measure, the outputs for the saem queries is shown below,

## Query 1:

```
**********
QUERY NO: 0
QUERY: ['deepfried', 'flatbread']

TOP 10 RANKINGS:

Document ID     Cosine Similarity with query
23               0.5106944347780581
92               0.3516205550719911
44               0.19027931230030726
30               0.11374068581147201
91               0.09448419581652645
3                0.07792131089792209
85               0.04718540515796432
41               0.04530916930921902
13               0.03842470377628093
99               0.0

TOP 3 RANKED DOCUMENTS:
Document ID: 23
Content:
 white pea mixed finely chopped onion tomato chilli coriander coconut tamarind juice make delicacy traditionally eaten luchi deepfried flatbread radha ballabhi stuffed deepfried flatbread

Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 44
Content:
 vadas crisp fried doughnut made black gram lentil spice like cumin seed green chilli crispy deepfried snack usually served coconut chutney sambhar

**********
```

## Query 2:

```
**********
QUERY NO: 1
QUERY: ['standing', 'waterfront', 'pichola']

TOP 10 RANKINGS:

Document ID     Cosine Similarity with query
0                0.19453401276366344
9                0.10534931275569903
10               0.0592797178941763
1                0.051747651729039834
99               0.0472817663549735
13               0.034636808349636894
98               0.0
97               0.0
96               0.0
95               0.0

TOP 3 RANKED DOCUMENTS:
Document ID: 0
Content:
 standing waterfront lake pichola gangori ghat stunning bagore ki haveli constructed 18th century amar chand badwa prime minister mewar kingdom bagore ki haveli wa private property till india independenc

Document ID: 9
Content:
 located heart lake pichola ethereal lake palace seems like mirage dream white floating brilliant blue lake every sunrise sunset palace seems melt molten gold shimmering water

Document ID: 10
Content:
 popular historical stopover tourist circuit teen murti house wa residence first prime minister india jawaharlal nehru lived 16 year death year 1964 house wa dedicated memorial building called teen murti

**********
```

## Query 3:

```
**********
QUERY NO: 2
QUERY: ['traditional', 'curry', 'pakoda', 'boiled', 'rice']

TOP 10 RANKINGS:

Document ID     Cosine Similarity with query
30               0.5834239175349033
92               0.19287168800513466
11               0.11334370681820805
96               0.08889698687547454
62               0.07676995639667716
34               0.07028489146281192
56               0.06696113452766383
35               0.06256645857088539
94               0.048602884969689564
50               0.045125709757016796

TOP 3 RANKED DOCUMENTS:
Document ID: 30
Content:
 kadhi traditional curdbased curry fried gram flour fritter called pakoda often eaten boiled rice roti indian flatbread

Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 11
Content:
 unique dish mirchi ka saalan spicy gravy green chilli cooked peanutandsesameseed curry coconut khus khus poppy seed also added curry flavour texture said dish wa one favourite mughal emperor akbar even

**********
```

Observations: Cosine similarity can very well handle the cons of euclidean distance. It can be clearly seen that the generated rankings are very relevant to the input query in all of the cases. Cosine similarity is also able to capture long documents which are relevant unlike euclidean distance. Using cosine similarity eliminates the need for normalization of the documents or query vectors.

**Q. Experiment with different variants of TF and IDF and not the changes in the term weights when different combinations were used.**

Four combinations of TF and IDF were used,

- Log Normalized TF and smooth IDF
- Log Normalized TF and probabilistic IDF
- Double Normalized-K TF and smooth IDF
- Double Normalized-K TF and probabilistic IDF

A reference document was chosen and the term weights were calculated in each of the cases. The output is shown below,

| Word | LN+Prob | LN+Smooth | DN+Prob | DN+Smooth |
|---|---|---|---|---|
| architecture | 3.17 | 3.459 | 1.506 | 1.643 |
| lady | 4.585 | 4.7 | 2.178 | 2.233 |
| constructed | 3.97 | 4.143 | 1.886 | 1.968 |
| stunning | 4.585 | 4.7 | 2.178 | 2.233 |
| renowned | 3.732 | 3.934 | 1.773 | 1.869 |
| 138 | 6.629 | 6.658 | 3.149 | 3.163 |
| jharokhas | 6.629 | 6.658 | 3.149 | 3.163 |
| ostentatious | 6.629 | 6.658 | 3.149 | 3.163 |
| glass | 3.97 | 4.143 | 1.886 | 1.968 |
| property | 6.629 | 6.658 | 3.149 | 3.163 |
| chand | 6.629 | 6.658 | 3.149 | 3.163 |
| around | 4.248 | 4.392 | 2.018 | 2.086 |
| decorated | 5.015 | 5.102 | 2.382 | 2.423 |
| assortment | 6.629 | 6.658 | 3.149 | 3.163 |
| lake | 4.585 | 4.7 | 2.178 | 2.233 |
| museum | 4.248 | 4.392 | 2.018 | 2.086 |
| minister | 5.015 | 5.102 | 2.382 | 2.423 |
| mansion | 6.629 | 6.658 | 3.149 | 3.163 |
| including | 3.338 | 3.598 | 1.585 | 1.709 |
| till | 4.585 | 4.7 | 2.178 | 2.233 |
| archway | 5.015 | 5.102 | 2.382 | 2.423 |
| pichola | 5.615 | 5.672 | 2.667 | 2.694 |
| intricate | 3.524 | 3.755 | 1.674 | 1.784 |
| rich | 5.015 | 5.102 | 2.382 | 2.423 |
| private | 5.615 | 5.672 | 2.667 | 2.694 |
| mewar | 5.615 | 5.672 | 2.667 | 2.694 |
| balcony | 6.629 | 6.658 | 3.149 | 3.163 |
| room | 4.248 | 4.392 | 2.018 | 2.086 |
| today | 3.732 | 3.934 | 1.773 | 1.869 |
| chamber | 5.015 | 5.102 | 2.382 | 2.423 |
| huge | 3.732 | 3.934 | 1.773 | 1.869 |
| ghat | 3.97 | 4.143 | 1.886 | 1.968 |

Observations: It can be seen that using log normalized TF generates higher values of term weights than using double normalized-K TF. Also it is observed that probabilistic IDF gives lower values of term weights compared to smooth IDF.

**Q. Use at least two different TF-IDF and report changes observed in ranked lists w.r.t. all queries.**

**First combination:** Log normalized TF + probabilistic IDF + cosine similarity.
Outputs:
Query 1:

```
********************************
QUERY NO: 0
QUERY: ['deepfried', 'flatbread']

TOP 10 RANKINGS:

Document ID     Euclidean distance with query
23               0.5096252228022501
92               0.3516165262672819
44               0.1916852893740819
30               0.11193276351934545
91               0.09276117092475589
3                0.07603810841195924
85               0.046108451821672 67
41               0.04437050023606348
13               0.03747416017247229
99               0.0

TOP 3 RANKED DOCUMENTS:
Document ID: 23
Content:
 white pea mixed finely chopped onion tomato chilli coriander coconut tamarind juice make delicacy traditionally eaten luchi deepfried flatbread radha ballabhi stuffed deepfried flatbread

Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 44
Content:
 vadas crisp fried doughnut made black gram lentil spice like cumin seed green chilli crispy deepfried snack usually served coconut chutney sambhar

********************************
```

Query 2:

```
********************************
QUERY NO: 1
QUERY: ['standing', 'waterfront', 'pichola']

TOP 10 RANKINGS:

Document ID     Euclidean distance with query
0                0.1947078133104736
9                0.10570618133637244
10               0.05915586903064 2405
1                0.051822632168479155
99               0.04707527751903984
13               0.03433508475081459
98               0.0
97               0.0
96               0.0
95               0.0

TOP 3 RANKED DOCUMENTS:
Document ID: 0
Content:
 standing waterfront lake pichola gangori ghat stunning bagore ki haveli constructed 18th century amar chand badwa prime minister mewar kingdom bagore ki haveli wa private property till india independ

Document ID: 9
Content:
 located heart lake pichola ethereal lake palace seems like mirage dream white floating brilliant blue lake every sunrise sunset palace seems melt molten gold shimmering water

Document ID: 10
Content:
 popular historical stopover tourist circuit teen murti house wa residence first prime minister india jawaharlal nehru lived 16 year death year 1964 house wa dedicated memorial building called teen mu

********************************
```

Query 3:

```
*********************************
QUERY NO: 2
QUERY: ['traditional', 'curry', 'pakoda', 'boiled', 'rice']

TOP 10 RANKINGS:

Document ID     Euclidean distance with query
30               0.5864074552284007
92               0.19284475009118296
11               0.113020524154725 41
96               0.08905418173274837
62               0.07676031911327674
34               0.06909561141657905
56               0.06672968245890121
35               0.06283098804171172
94               0.046225473572096165
50               0.04476260250616293

TOP 3 RANKED DOCUMENTS:
Document ID: 30
Content:
 kadhi traditional curdbased curry fried gram flour fritter called pakoda often eaten boiled rice roti indian flatbread

Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 11
Content:
 unique dish mirchi ka saalan spicy gravy green chilli cooked peanutandsesameseed curry coconut khus khus poppy seed also added curry flavour texture said dish wa one favourite mughal emperor akbar

*********************************
```

Observations: In case of this combination the rankings generated by the standard TF-IDF and this variant is exactly the same for all three queries. The cosine similarity values are also almost similar upto 2 decimal places. Hence very minute changes are observed in terms of the similarity measure but the ranking remains the same.

**Second combination:** Double Normalized-K TF + Smooth IDF + Euclidean distance.

Outputs:

Query 1:

```
*********************************
QUERY NO: 0
QUERY: ['deepfried', 'flatbread']

TOP 10 RANKINGS:

Document ID     Euclidean Distance with query
92               95.83992353085887
96               96.06844676922334
3                96.07056332927066
21               96.08276694562484
23               96.1500357254336
33               96.20174365158046
75               96.25148723001715
73               96.27695276310855
87               96.30515777108285
9                96.32301083799965

TOP 3 RANKED DOCUMENTS:
Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 96
Content:
 delicious curry prepared gram flour dish gatte steamed gram flour dumpling cooked spicy curdbased gravy

Document ID: 3
Content:
 essentially fried roti flatbread delicious filling kathi roll roll wellloved snack kolkata paneer mutton chicken may used filling roll along egg

*********************************
```

## Query 2:

```
********************************
QUERY NO: 1
QUERY: ['standing', 'waterfront', 'pichola']

TOP 10 RANKINGS:

Document ID     Euclidean Distance with query
92               96.10042185645791
96               96.1847353649432
21               96.19903823064094
3                96.2200345579906
33               96.31787131253192
9                96.36150288433501
75               96.36755494751961
73               96.39298981732487
87               96.42116088227843
1                96.44040877863681

TOP 3 RANKED DOCUMENTS:
Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 96
Content:
 delicious curry prepared gram flour dish gatte steamed gram flour dumpling cooked spicy curdbased gravy

Document ID: 21
Content:
 special marinade used entire goat cooked charcoal fire make delicacy wedding usually occasion much meat cooked one go

********************************
```

## Query 3:

```
********************************
QUERY NO: 2
QUERY: ['traditional', 'curry', 'pakoda', 'boiled', 'rice']

TOP 10 RANKINGS:

Document ID     Euclidean Distance with query
92               96.01087786248969
96               96.16438634032204
21               96.2477487297064
3                96.26873443323791
33               96.36652174491844
75               96.41618031015122
73               96.44160235582606
30               96.46423456426419
87               96.46975922496436
9                96.48758183573072

TOP 3 RANKED DOCUMENTS:
Document ID: 92
Content:
 deepfried flatbread made wheat generous amount ghee clarified butter eaten curry especially fish curry

Document ID: 96
Content:
 delicious curry prepared gram flour dish gatte steamed gram flour dumpling cooked spicy curdbased gravy

Document ID: 21
Content:
 special marinade used entire goat cooked charcoal fire make delicacy wedding usually occasion much meat cooked one go

********************************
```

Observations: Here it can be seen that the rankings have changed w.r.t the standard TF-IDF weighting but the top ranked documents are still not relevant due to the use of classic euclidean distance measure. The rankings changed because of the use of double normalized-K TF. But the drawback of shooter documents being favored is still not resolved in this case. It may be resolved by normalizing all the vectors in the vector space.

THANK YOU