

IR

# Assignment 2

---

Akhil Majahan (MT20107)

Anjali (MT20082)

Prabal Jain (MT20115)

Shradha Sabhlok(MT20069)

## Q1. Analysis on Different Phrase Queries.

Preprocessing :

```
[10] test_before = "DING DONG! The melancholy tintinabulation"

print('Before Preprocessing : {}'.format(test_before))
print('After Preprocessing : {}'.format(preprocess(test_before, False)))

Before Preprocessing : DING DONG! The melancholy tintinabulation
After Preprocessing : ding dong melancholy tintinabulation
```

Test Query : “Good day”

“Without Lemmatization”

```
[19] # Length of documents retrived
print("Number of Documents retrived are :",len(result))
```

Number of Documents retrived are : 19

```
[20] for id in list(result.keys()):
    print(dataset[id][0].split('/')[6:])
```

```
['stories', '13chil.txt']
['stories', 'aesopa10.txt']
['stories', 'brain.damage']
['stories', 'breaks2.asc']
['stories', 'enchdup.hum']
['stories', 'fantasy.hum']
['stories', 'fantasy.txt']
['stories', 'fic5']
['stories', 'forgotte']
['stories', 'history5.txt']
['stories', 'horswolf.txt']
['stories', 'hound-b.txt']
['stories', 'mazarin.txt']
['stories', 'melissa.txt']
['stories', 'outcast.dos']
['stories', 'sick-kid.txt']
['stories', 'startrek.txt']
['stories', 'superg1']
['stories', 'SRE', 'srex.txt']
```

With Lemmatization :

```
[36] # Length of documents retrived
      print("Number of Documents retrived are :",len(result))
```

```
Number of Documents retrived are : 21
```

```
▶ for id in list(result.keys()):
    print(dataset[id][0].split('/')[6:])
```

```
↳ ['stories', '13chil.txt']
['stories', 'aesop11.txt']
['stories', 'aesopa10.txt']
['stories', 'brain.damage']
['stories', 'breaks2.asc']
['stories', 'bruce-p.txt']
['stories', 'enchdup.hum']
['stories', 'fantasy.hum']
['stories', 'fantasy.txt']
['stories', 'fic5']
['stories', 'forgotte']
['stories', 'history5.txt']
['stories', 'horswolf.txt']
['stories', 'hound-b.txt']
['stories', 'mazarin.txt']
['stories', 'melissa.txt']
['stories', 'outcast.dos']
['stories', 'sick-kid.txt']
['stories', 'startrek.txt']
['stories', 'superg1']
['stories', 'SRE', 'srex.txt']
```

Test Query : “Stay on watch on the bridge”.

```
[19] # Length of documents retrived
      print("Number of Documents retrived are : " ,len(result))
```

```
Number of Documents retrived are : 1
```

```
[20] for id in list(result.keys()):
      print(dataset[id][0].split('/')[6:])
```

```
['stories', '6ablemen.txt']
```

```
[21] result
```

```
{15: [33]}
```



```
print_doc(15)
```

```
('/content/gdrive/My Drive/InformationRetrieval/Assignment_1/stories/6ablemen.txt',
SIX ABLE MEN
```

Once upon a time there lived a young soldier named Martin who had enlisted in the royal army to fight a war. The war was long but victorious and when the King abandoned the enemy's territory and returned with his troops to the homeland, he left Martin to guard the only bridge on the river that separated the two nations.

"Stay on watch on the bridge," the King ordered. "Don't let any enemy soldier go by." Days and then months passed, and the soldier kept his watch on the bridge. He survived by asking the passers-by for food and, after two years, thought that the authorities had probably forgotten him. He then headed towards the capital, where he would ask the King for all his back pay. His pockets were empty and his only possessions were a pipe, a bit of tobacco and his sword.

## Question 2 : Scoring and Term Weighting

Output for Jaccard Coefficient:

```
1] ▶ MI
jaccard_coefficient = get_jaccard_values(query)

0.0
0.0
0.0005737234652897303
0.002544529262086514
0.0
0.0
0.0
0.0
0.0
0.0
0.006134969325153374
```

Top 5 values of Jaccard Coefficient are retrieved and their corresponding documents are named in the code along with their Jaccard Coefficients wrt the input query.

```
[0.004878048780487805, 0.004962779156327543, 0.004962779156327543, 0.0051813471502590676, 0.006134969325153374]
0.006134969325153374 and index is 28
0.0051813471502590676 and index is 211
0.004962779156327543 and index is 386
0.004962779156327543 and index is 387
0.004878048780487805 and index is 235

List of Top Documents wrt Jaccard Coefficient for i/p query is:
aircon.txt with Jaccard Similarity Value: 0.006134969325153374
graymare.txt with Jaccard Similarity Value: 0.0051813471502590676
social.vikings with Jaccard Similarity Value: 0.004962779156327543
socialvikings.txt with Jaccard Similarity Value: 0.004962779156327543
hotline4.txt with Jaccard Similarity Value: 0.004878048780487805
```

As can be seen in the above snippet, for the input query “telephone,paved, roads”, **the top 5 documents are aircon.txt, graymare.txt, social.vikings, socialvikings.txt, hotline4.txt** and the respective Jaccard Coefficient values is also mentioned in the output.

## Result for binary tf-

```
Matching Score

Query: THE CRAB AND THE HERON

['crab', 'heron']

[122, 26, 418, 270, 169]
9.406564833939129
9.406564833939129
4.356708826689592
4.356708826689592
4.356708826689592
['crabhern.txt', 'aesop11.txt', 'timem.hac', 'long1-3.txt', 'fgoose.txt']
```

## Result for raw count -

```
Matching Score

Query: THE CRAB AND THE HERON

['crab', 'heron']

[122, 26, 418, 270, 169]
83.96593632489221
31.19010896738709
13.070126480068776
8.713417653379183
4.356708826689592
['crabhern.txt', 'aesop11.txt', 'timem.hac', 'long1-3.txt', 'fgoose.txt']
```

## Result for Tf-

```
Matching Score

Query: THE CRAB AND THE HERON

['crab', 'heron']

[122, 26, 270, 418, 169]
0.41362530209306514
0.0016366746585185015
0.001309303929884175
0.0008390118423461789
0.000632598929387192
['crabhern.txt', 'aesop11.txt', 'long1-3.txt', 'timem.hac', 'fgoose.txt']
```

### Result for Log Normalisation -

```
Matching Score

Query: THE CRAB AND THE HERON

['crab', 'heron']

[122, 26, 418, 270, 169]
21.54259923161554
11.978057375992861
6.039680879481035
4.786333855150008
3.0198404397405176
['crabhern.txt', 'aesop11.txt', 'timem.hac', 'long1-3.txt', 'fgoose.txt']
```

### Result for Double Normalisation -

```
Matching Score

Query: THE CRAB AND THE HERON

['crab', 'heron']

[122, 26, 270, 418, 169]
8.901579233214175
4.776156503341964
2.2296098113058496
2.210867165782778
2.2099247671613873
['crabhern.txt', 'aesop11.txt', 'long1-3.txt', 'timem.hac', 'fgoose.txt']
```

**COSINE SIMILARITY :** We have done this for all the applied Weighting Schemes for the tf-idf and obtained results are logged in the code.

For Binary Weighting Scheme:

```
List of Top Documents wrt Cosine Similarity for i/p query(Binary TF Weighting) is:
crabhern.txt with Cosine Similarity Value: 0.2401234519815826
aesop11.txt with Cosine Similarity Value: 0.025439293354229037
long1-3.txt with Cosine Similarity Value: 0.0211669121083841
fgoose.txt with Cosine Similarity Value: 0.01770831819721052
timem.hac with Cosine Similarity Value: 0.013841932742447044
```

For Raw Count Weighting Scheme:

```
List of Top Documents wrt Cosine Similarity for i/p query(Raw count) is:
crabhern.txt with Cosine Similarity Value: 0.7973139054443423
aesop11.txt with Cosine Similarity Value: 0.018439642087588422
timem.hac with Cosine Similarity Value: 0.011905221345725029
long1-3.txt with Cosine Similarity Value: 0.010776157800016615
fgoose.txt with Cosine Similarity Value: 0.006676890533805383
```

For Term Frequency Scheme:

```
List of Top Documents wrt Cosine Similarity for i/p query is:
crabhern.txt with Cosine Similarity Value: 0.7973139054443426
aesop11.txt with Cosine Similarity Value: 0.018439642087588456
timem.hac with Cosine Similarity Value: 0.011905221345725024
long1-3.txt with Cosine Similarity Value: 0.010776157800016617
fgoose.txt with Cosine Similarity Value: 0.006676890533805379
```


For Log Normalization Scheme:

```
List of Top Documents wrt Cosine Similarity for i/p query(Log Normalization) is:
crabhern.txt with Cosine Similarity Value: 0.5607367318281735
aesop11.txt with Cosine Similarity Value: 0.03016404874411095
long1-3.txt with Cosine Similarity Value: 0.023900241380333767
timem.hac with Cosine Similarity Value: 0.019228825736402343
fgoose.txt with Cosine Similarity Value: 0.012887410111712457
```

For Double Normalization Scheme:

```
List of Top Documents wrt Cosine Similarity for i/p query(Double Normalization) is:
crabhern.txt with Cosine Similarity Value: 0.3808980579800617
aesop11.txt with Cosine Similarity Value: 0.02557373919250654
long1-3.txt with Cosine Similarity Value: 0.02119659412756276
fgoose.txt with Cosine Similarity Value: 0.01753623169044474
timem.hac with Cosine Similarity Value: 0.013919207191540366
```





## Pros and cons of each approach-

### Jaccard Similarity

#### Pros

- Used in applications where binary or binarized data are used
- Used to compare set of patterns
- It is good for cases where duplication does not matter

#### Cons

- Jaccard index is highly influenced by the size of the data. Large datasets can have a big impact on the index as it could significantly increase the union whilst keeping the intersection similar.

### TF-IDF Scheme

#### Pros

- Easy to compute
- Easy to compute similarity between two documents
- We can extract most descriptive terms using this scheme
- TF is good for text similarity in general, but TF-IDF is good for search query relevance.

#### Cons

- Term ordering is not considered
- Cannot capture position in text or semantics.

### Cosine Similarity

#### Pros

- It is low-complexity, especially for sparse vectors
- The cosine similarity is beneficial because even if the two similar documents are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them
- cosine similarity is good for cases where duplication matters while analyzing text similarity

Page 10 of 10

- ### Question 3 : Ranked-Information Retrieval and Evaluation

A file with the name “**max\_DCG.txt**” is created in order of the max DCG. In the given datafile, there are 59 zero’s, 26 one’s, 17 two’s and 1 three as relevance scores for **qid:4**. So the total number of files that can be made are:  $1! * 17! * 26! * 59!$  = 19893497375938370599826047614905329896936840170566570588205180312704857992695193482412686565431050240000000000000000000000000000

**a. At 50**

For gid:4 query-url pairs in the given file, following values are obtained:

9

**Q. Assume a model that simply ranks URLs on the basis of the value of feature 75 (sum of TF-IDF on the whole document) i.e. the higher the value, the more relevant the URL. Assume any non zero relevance judgment value to be relevant. Plot a Precision-Recall curve for query “qid:4”.**

The precision and recall values are plotted using plot function from matplotlib library and the plot is shown below:

