CSE508 IR Assignment 2
Group 41

Q1.

Total files used(467)

Positional index implementation:

The output shown below basically contains all the unique words with their positional index lin the form like (documentID: position1, position2,....)

	disengaging	pentration	royalship	phalanxes	invades	interrogating	subduing	navterm	bov	wingmen	latching	shockwace	nalsi	dissatisfaction
0	[[466, {1773}]]	[[466, {1673}]]	[[466, {1668, 1706, 1719, 1723, 1755}]]	[[466, {1616}]]	[[466, {1510}]]	[[466, {1475}]]	[[466, {1467}]]	[[466, {1361}]]	[[466, {1339, 1332}]]	[[466, {1298}]]	[[466, {1277}]]	[[466, {1176}]]	[[466, {1144, 1098, 1164}]]	[[466, {862}]]

Output for frequency of each word it is coming something like this:

trees	pretend	field	air	dry	inhale	arms	stretch	trampled	upon	walk	outsideand	go	to	american	great	feels	it
101	24	100	196	79	4	145	36	9	246	142	1	314	78	55	238	20	63

We tried query "good day" and **got 21 matches:**

```
good day
['good', 'day']
iniword good
inimatches [(0, 1377), (0, 10), (0, 1179), (1, 732), (3, 33), (5, 177), (5, 211), (5, 100), (13, 136), (13, 2952), (13, 1548), (13, 4108
['day']
Number of Document Mathced are: {129, 131, 13, 285, 287, 163, 168, 430, 303, 51, 190, 192, 193, 327, 328, 201, 207, 464, 98, 229, 120}
['fic5', 'aesopa10.txt', 'forgotte', 'sick-kid.txt', '13chil.txt', 'aesop11.txt', 'superg1', 'enchdup.hum', 'melissa.txt', 'history5.txt
total document matches 21
```

Q2 : Scoring and Term-Weighting

2.1 Jaccard Coefficient:

Here we have have query "**good day**" then we find jaccard coefficient the top 5 documents are shown in the highlighted part on below image.

```
Enter the query for fetching top 5 docs based on jaccard coefficientgood day
input query tokens are ['good', 'day']
jaccard coefficient of docs {436: 0.02, 385: 0.017543859649122806, 15: 0.01612903225806
top 5 relevant documents based on the value of the Jaccard coefficient are:
[436, 385, 15, 100, 285]
quarter.c16
blasters.fic
cameloto.hum
aminegg.txt
foxnstrk.txt
```

2.2 TF-IDF Matrix:

a. Find the Tf-Idf for all 5 variants

i. Raw Count Variant:

In this we use input query is -> "good day"

```
Enter the String   : good day
[96, 407]
query vector  1.0
query vector  1.0
query vector  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
{86: 186.68417688616, 448: 136.05855374463943, 333: 123.2884655845475, 127: 110.07
Top  5  Documents based on  Raw_Count  tf-idf are [86, 448, 333, 127, 308]
gulliver.txt
vgilante.txt
hound-b.txt
outcast.dos
aesop11.txt
```

ii. Term Frequency Variant:

In this we use input query is -> "good day"

```
[96, 407]
query vector  1.0
query vector  1.0
query vector  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
{150: 0.0358616731276486, 13: 0.03293560267688336, 385: 0.026740294280144364, 378: 0
Top  5  Documents based on  termfrequency  tf-idf are [150, 13, 385, 378, 332]
blossom.pom
contrad1.hum
blasters.fic
clevdonk.txt
horswolf.txt
```

iii. Log Normalization Variant:

In this we use input query is -> “good day”

```
[96, 407]
query vector  1.0
query vector  1.0
query vector  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
{86: 8.549192330562068, 333: 7.802832021395227, 448: 7.610668085368812, 127: 7.594
Top  5  Documents based on  Logarithmic  tf-idf are [86, 333, 448, 127, 308]
gulliver.txt
hound-b.txt
vgilante.txt
outcast.dos
aesop11.txt
```

iv. Double Normalization Variant:

In this we use input query is -> “good day”

```
[96, 407]
query vector  1.0
query vector  1.0
query vector  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
{398: 1.514227760233382, 413: 1.514227760233382, 184: 1.4306995841199317, 188: 1.3352
Top  5  Documents based on  Double  tf-idf are [398, 413, 184, 188, 98]
pepdegener.txt
pepsi.degenerat
7voysinb.txt
jaynejob.asc
yukon.txt
```

v. Binary Variant:

In this we use input query is -> “good day”

```
[96, 407]
query vector  1.0
query vector  1.0
query vector  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
{0: 1.8524058234335237, 2: 1.8524058234335237, 4: 1.8524058234335237, 5: 1.852405823433
Top  5  Documents based on  Binary  tf-idf are [0, 2, 4, 5, 6]
blind.txt
partya.txt
sight.txt
tree.txt
beyond.hum
```

2.3.Cosine Similarity:

a. Top 5 documents are fetched on the basis of cosine score using all 5 weighting schemes.

```
Top  5  Documents found on basis of    tf-idf using log method are [5, 98, 150, 49, 88]
0    bestwish
Name: 5, dtype: object
0    horswolf.txt
Name: 98, dtype: object
0    mouslion.txt
Name: 150, dtype: object
0    pepdegener.txt
Name: 49, dtype: object
0    pepsi.degenerat
Name: 88, dtype: object
Top  5  Documents found on basis of    tf-idf using raw method are [269, 49, 88, 229, 442]
0    blossom.pom
Name: 269, dtype: object
0    pepdegener.txt
Name: 49, dtype: object
0    pepsi.degenerat
Name: 88, dtype: object
0    brain.damage
Name: 229, dtype: object
0    contrad1.hum
Name: 442, dtype: object
Top  5  Documents found on basis of    tf-idf using binary methodare [5, 98, 416, 150, 353]
0    bestwish
Name: 5, dtype: object
0    horswolf.txt
Name: 98, dtype: object
0    elveshoe.txt
Name: 416, dtype: object
0    mouslion.txt
Name: 150, dtype: object
0    aminegg.txt
Name: 353, dtype: object
```

```
Top  5  Documents found on basis of   tf-idf using double method are [5, 98, 150, 82, 416]
0    bestwish
Name: 5, dtype: object
0    horswolf.txt
Name: 98, dtype: object
0    mouslion.txt
Name: 150, dtype: object
0    blasters.fic
Name: 82, dtype: object
0    elveshoe.txt
Name: 416, dtype: object
Top  5  Documents found on basis of    tf-idf using term are [269, 49, 88, 229, 442]
0    blossom.pom
Name: 269, dtype: object
0    pepdegener.txt
Name: 49, dtype: object
0    pepsi.degenerat
Name: 88, dtype: object
0    brain.damage
Name: 229, dtype: object
0    contrad1.hum
Name: 442, dtype: object
```

Analysis of Scoring Schemes
Pro and Cons of each Scoring Schemes:

	Pros	**Cons**
Jaccard coefficient	Better Result where duplication or repetition of words does not matter.	Term frequency is not considered so it doesn't consider rare terms in a collection.
TF-IDF Matrix	Easy to compute the similarity b/w 2 different documents. Rare terms are more informative than frequent terms.	It is based on the bag-of-words (BoW) model, therefore it does not capture position in text, semantics, co-occurrences in different-different documents, etc.
Cosine Similarity	Smaller angles between documents have higher similarity that helps in clustering and classification between the documents. Basically used to Classify the documents.	The cosine similarity looks at "directional similarity" rather than magnitudinal differences.**cosine distance** is only concerned with the orientation of two points and not with their exact placement. This means that **cosine distance** is much less effected by **magnitude**, or how large your numbers are.

Q3.

Discounted cumulative gain implementation :

DGC measures the ranking quality and the competentivity of the search algorithms.

The output shown here is the data shared and the file is read using the pandas dataframe.

```
7 q3final_df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	qid:4	1:3	2:0	3:2	4:0	5:3	6:1	7:0	8:0.666667	9:0	10:1	11:999	12:0	13:110	14:5	15:1114	16:14.976692	17:28.949002	1
1	0	qid:4	1:3	2:0	3:3	4:0	5:3	6:1	7:0	8:1	9:0	10:1	11:1561	12:2	13:34	14:10	15:1607	16:14.976692	17:28.949002	1
2	0	qid:4	1:3	2:0	3:2	4:0	5:3	6:1	7:0	8:0.666667	9:0	10:1	11:1029	12:0	13:110	14:6	15:1145	16:14.976692	17:28.949002	1
3	0	qid:4	1:3	2:0	3:3	4:0	5:3	6:1	7:0	8:1	9:0	10:1	11:1786	12:0	13:30	14:6	15:1822	16:14.976692	17:28.949002	1
4	1	qid:4	1:3	2:0	3:3	4:0	5:3	6:1	7:0	8:1	9:0	10:1	11:725	12:0	13:35	14:6	15:766	16:14.976692	17:28.949002	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
98	0	qid:4	1:3	2:0	3:2	4:0	5:3	6:1	7:0	8:0.666667	9:0	10:1	11:227	12:0	13:9	14:10	15:246	16:14.976692	17:28.949002	1
99	1	qid:4	1:3	2:0	3:3	4:2	5:3	6:1	7:0	8:1	9:0.666667	10:1	11:406	12:1	13:11	14:9	15:427	16:14.976692	17:28.949002	1
100	2	qid:4	1:2	2:0	3:2	4:0	5:2	6:0.666667	7:0	8:0.666667	9:0	10:0.666667	11:656	12:0	13:9	14:4	15:669	16:14.976692	17:28.949002	1
101	1	qid:4	1:2	2:0	3:2	4:0	5:2	6:0.666667	7:0	8:0.666667	9:0	10:0.666667	11:1309	12:0	13:9	14:4	15:1322	16:14.976692	17:28.949002	1
102	0	qid:4	1:3	2:0	3:2	4:0	5:3	6:1	7:0	8:0.666667	9:0	10:1	11:399	12:5	13:13	14:9	15:426	16:14.976692	17:28.949002	1

103 rows × 139 columns

2)

We have rearranged query-url pairs in order of max DCG and made afile with name **q3DCG.csv**.

The output shown below is the maximum number of files after rearranging the query-urls of qid:4.

```
print("Number of files after rearranging the query-url pairs in order of max DCG for qid:4 is ",result)
```

```
rder of max DCG for qid:4 is  19893497375938370599826047614905329896936840170566570588205180312704857992695193482412686565431050240000000000
```

3)

The nDGC for 50 and the whole document are shown below.

```
Ques 3 part 3 i) nDCG at 50: 0.35612494416255847
Ques 3 part 3 ii) nDCG for whole dataset: 0.5784691984582591
```

4)

Below is the plot of Precision and Recall curve for query “qid:4”. The recall values are on the x axis and Precision values on the y axis.

Conclusion:

The plot grows straight exponentially between 0 and 0.2 Recall values and reaches value above 0.5 recall. After that the precision remains between 0.4 and 0.55 as the recall values span over 0.2 and 1.0