Homework 4 part 2

Objective

The objective of this assignment is to extend your indexing to store tf*idf weights and build a common-line retrieval engine that efficiently processes and ranks queries using an inverted file.

Approach

I made no change to the indexing code, the same hash function that I use for assign index, is used for querying. They query syntax is “query <term1> <term2> … <term n>”. The data structure use for the accumulator is a LinkedList for future algorithm exploration. The query processing algorithm find all the associated posting record for each term, summing up all the weight tf idf weight per document and displaying it in descending order.

File size for “tinyfiles” test case

	Dict	**Post**	**Map**
NumRecords from wc -l	30	11	4
Filesize from ls -l	1950	220	200
RecordSize: Filesize/NumRecords	65	20	50

My formula for idf is:

$$1 + log_2(N/df_t)$$

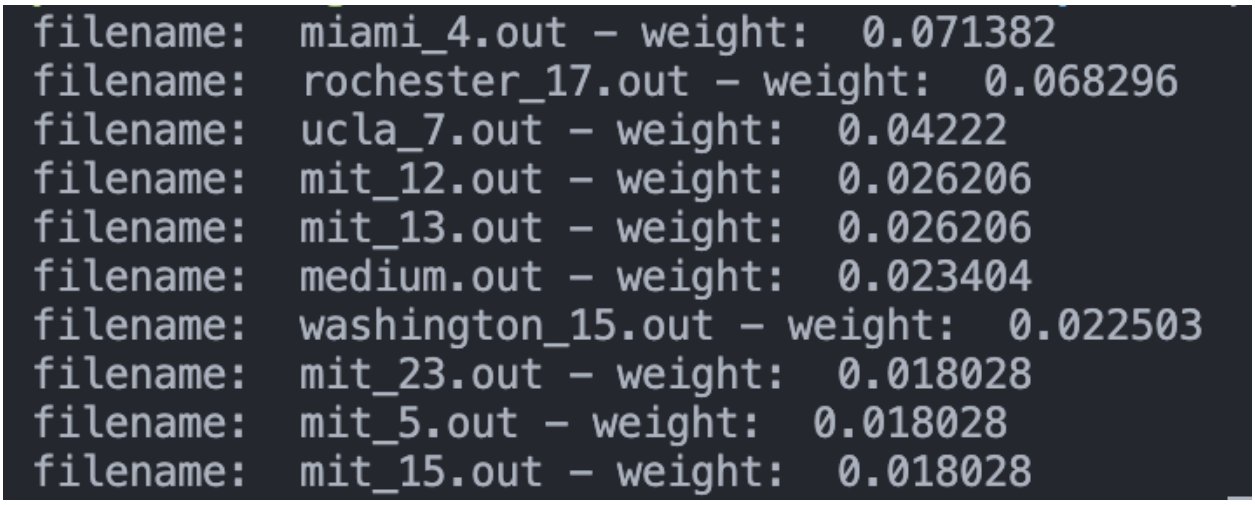
- N = number of document in collection
- df_t = frequency of a term in that document

Fill in the table below with values calculated using a calculator (not what is in the file):

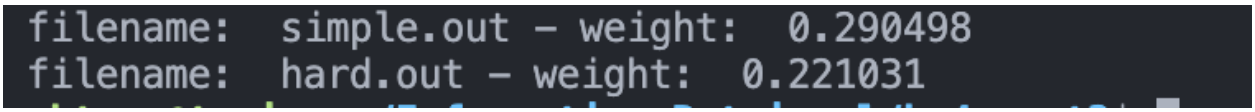
Term	NumDocs (from dict)	idf value
dog	2	2
quickly	1	3

Queries

- **20**



- **Gauch**



- **The**: none return
- **abracadabra:** none return
- **HoNoRaRy**

```
filename:  harvard_6.out - weight:  0.016451
filename:  nd_30.out - weight:  0.00556
filename:  northwestern_2.out - weight:  0.003858
filename:  cmu_7.out - weight:  0.003689
filename:  harvard_30.out - weight:  0.003649
filename:  uci_20.out - weight:  0.00364
filename:  harvard_27.out - weight:  0.003426
filename:  harvard_4.out - weight:  0.002864
filename:  harvard_17.out - weight:  0.002493
filename:  harvard_25.out - weight:  0.002164
```

- **doctorate!!!:** no result
- **doctorate**

```
filename:  rochester_17.out - weight:  0.0217
filename:  ucla_3.out - weight:  0.004715
filename:  msu_15.out - weight:  0.003904
filename:  nd_30.out - weight:  0.003152
filename:  gatech_14.out - weight:  0.001974
filename:  jhu_15.out - weight:  0.001675
filename:  rice_29.out - weight:  0.00135
filename:  washington_5.out - weight:  0.001336
filename:  msu_7.out - weight:  0.001318
filename:  pitt_2.out - weight:  0.001004
```

- **honorary doctorate**

```
filename:  rochester_17.out - weight:  0.0217
filename:  harvard_6.out - weight:  0.016451
filename:  nd_30.out - weight:  0.008712
filename:  ucla_3.out - weight:  0.004715
filename:  msu_15.out - weight:  0.003904
filename:  northwestern_2.out - weight:  0.003858
filename:  cmu_7.out - weight:  0.003689
filename:  harvard_30.out - weight:  0.003649
filename:  uci_20.out - weight:  0.00364
filename:  harvard_27.out - weight:  0.003426
```

	0.html	1.html	2.html	3.html
Num_tokens	4	5	8	3
Freq(dog)	1	1	0	0
Rtf (dog)	0.25	0.20	0	0
Freq(quickly)	1	0	0	0
Rtf(quicly)	0.25	0	0	0
rtf*idf(dog)	0.50	0.4	0	0
rtf*idf(quickly)	0.75	0	0	0
Post wt (dog)	0.50	0.40	0	0
Post wt (quickly)	0.75	0	0	0
Post wt(dog)+wt(quickly)	1.25	0.40	0	0

Typescript

```
Script started on 2024-11-11 20:16:14-0600
groups: cannot find name for group ID 762800513
]0;phtran@turing: ~/Information-Retrieval/hw4-part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$ exit
exit
python3 SearchEngine.py query quickly
[4Pdog
filename: 0.out - weight: 0.5
filename: 1.out - weight: 0.4
]0;phtran@turing: ~/Information-Retrieval/hw4-part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$ python3 SearchEngine.py query dog[K[K[Kquickly
filename: 0.out - weight: 0.75
]0;phtran@turing: ~/Information-Retrieval/hw4-part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$ python3 SearchEngine.py query quickly[K[K[K[K[K[K[K[Kdoc[Kg quickly
filename: 0.out - weight: 1.25
filename: 1.out - weight: 0.4
]0;phtran@turing: ~/Information-Retrieval/hw4-part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$ python3 SearchEngine.py query dog
```

```
quickly[K[K[K[K[K[K[K[K[K[K[K[K[Kfish
]0;phtran@turing: ~/Information-Retrieval/hw4-
part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$
python3 SearchEngine.py query fish[K[K[K[K[Kthe
]0;phtran@turing: ~/Information-Retrieval/hw4-
part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$
python3 SearchEngine.py query the dog
filename: 0.out - weight: 0.5
filename: 1.out - weight: 0.4
]0;phtran@turing: ~/Information-Retrieval/hw4-
part2[01;32mphtran@turing[00m:[01;34m~/Information-Retrieval/hw4-part2[00m$
exit
exit

Script done on 2024-11-11 20:16:43-0600
```

Efficiency

- let n be the number of terms
- let k be the number of posting per terms
- let n_d be the number of record in the dict file

The time complexity for the worst case is:

$$O(n * n_d * k)$$

The time complexity for the best case is:

$$O(n * k)$$

Timings

query	time
razorbacks	0.79 ms
razorbacks college academics apply	141.40 ms