

Task 1

Subtask 1

$$w_{t,d} = (1 + \log tf_{t,d}) * \log \frac{N}{df_t}$$

- $w_{pens,d_1} = (1 + \log 1) * \log \frac{3}{1} \approx 1, 1$
- $w_{pens,d_2} = \text{not in the doc}$
- $w_{pens,d_3} = \text{not in the doc}$
- $w_{write,d_1} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{write,d_2} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{write,d_3} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{on,d_1} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{on,d_2} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{on,d_3} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{paper,d_1} = (1 + \log 2) * \log \frac{3}{3} = 0$
- $w_{paper,d_2} = \text{not in the doc}$
- $w_{paper,d_3} = (1 + \log 1) * \log \frac{3}{3} = 0$
- $w_{pencils,d_1} = \text{not in the doc}$
- $w_{pencils,d_2} = (1 + \log 1) * \log \frac{3}{1} \approx 1, 1$
- $w_{pencils,d_3} = \text{not in the doc}$
- $w_{envelope,d_1} = \text{not in the doc}$
- $w_{envelope,d_2} = (1 + \log 1) * \log \frac{3}{1} \approx 1, 1$
- $w_{envelope,d_3} = \text{not in the doc}$
- $w_{ballpens,d_1} = \text{not in the doc}$
- $w_{ballpens,d_2} = \text{not in the doc}$
- $w_{ballpens,d_3} = (1 + \log 1) * \log \frac{3}{1} \approx 1, 1$

Terms	d_1	d_2	d_3
pens	1.1	0.0	0.0
write	0.0	0.0	0.0
on	0.0	0.0	0.0
paper	0.0	0.0	0.0
pencils	0.0	1.1	0.0
envelope	0.0	1.1	0.0
ballpens	0.0	0.0	1.1

$$\vec{d}_1 = (1.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)$$

$$\vec{d}_2 = (0.0, 0.0, 0.0, 0.0, 1.1, 1.1, 0.0)$$

$$\vec{d}_3 = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.1)$$

Subtask 2

$$w_{ballpens,q} = (1 + \log 1) * \log \frac{3}{1} \approx 1, 1$$

$$w_{envelope,q} = (1 + \log 1) * \log \frac{3}{1} \approx 1, 1$$

$$\vec{q} = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.1, 1.1)$$

$$SIM(\vec{q}, \vec{d}_1) = \frac{\vec{q} * \vec{d}_1}{|\vec{q}| * |\vec{d}_1|} = 0$$

$$SIM(\vec{q}, \vec{d}_2) = \frac{\vec{q} * \vec{d}_2}{|\vec{q}| * |\vec{d}_2|} = \frac{1.21}{49} \approx 0.025$$

$$SIM(\vec{q}, \vec{d}_3) = \frac{\vec{q} * \vec{d}_3}{|\vec{q}| * |\vec{d}_3|} = \frac{1.21}{49} \approx 0.025$$

Rank	Doc	SIM
1	2	0.025
1	3	0.025
2	1	0

Task 2

Step 0: Information need: Does the result match the query?(match/not match)

Step 1: Search on google with query "computer not boot os x" and record the top 20 results.
the top 20 results:

1. What to Do When Your Mac Won't Start Up - How-To Geek
(<https://www.howtogeek.com/217158/what-to-do-when-your-mac-won't-start-up/>)
2. How to fix a Mac Or MacBook that won't Turn On Or Boot Up ...
(<https://www.macworld.co.uk/how-to/wont-turn-on-boot-3423817/>)
3. My Mac Won't Start or Boot: How To Fix White Screen
(<https://appletoolbox.com/mac-fix-white-screen/>)
4. 'Restart in OS X' Not Working! Boot Camp
(<https://developer.apple.com/forums/thread/79001>)
5. Mac Won't Turn On? How to Fix It and Make It Boot
(<https://www.makeuseof.com/tag/mac-wont-boot-step-step-guide-waking/>)
6. Mac won't boot/turn on after macOS Big Sur update - iBoysoft
(<https://iboysoft.com/mac-data-recovery/mac-computer-wont-boot-after-macos-update.html>)
7. [Solved] Mac Won't Boot into Recovery Mode - Recoverit
(<https://recoverit.wondershare.com/mac-tips/cannot-boot-into-recovery-mode-mac.html>)
8. Mac won't boot YouTube
(<https://www.youtube.com/watch?v=snm7p23K2E4>)
9. MacBook won't boot up? Here's how to fix the problem - Digital ...
(<https://www.digitaltrends.com/computing/apple-macbook-problems-starting-up/>)
10. Mac White Screen: How To Fix Your Mac That Won't Turn On
(<https://macreports.com/mac/>)
11. Mac OS X system does not boot correctly - DisplayLink Support
(<https://support.displaylink.com/knowledgebase/articles/589233-mac-os-x-system-does-not-boot-correctly>)
12. What to do when Mac OS X Yosemite won't boot - Macintosh ...
(<https://www.macintoshhowto.com/osx/what-to-do-when-mac-os-x-wont-boot.html>)
13. MacBook Air/Pro won't boot after macOS update, how to fix ...
(<https://techengage.com/macbook-air-pro-wont-boot-after-macos-update-how-to-fix/>)
14. Holding down the option key isn't allowing me to select ...
(<https://apple.stackexchange.com/questions/99289/holding-down-the-option-key-isnt-allowing-me-to-select-windows-os-x-on-startup>)
15. What to do when a Mac won't boot to Safe Mode - CNET
(<https://www.cnet.com/news/what-to-do-when-a-mac-wont-boot-to-safe-mode/>)
16. How to fix a Mac that won't start — TechRadar
(<https://www.techradar.com/how-to/computing/how-to-fix-a-mac-that-won-t-start-1319481>)
17. Mac SSD not Booting? Try 5 Fixes to Boot up Your Mac — by ...

(<https://medium.com/@kepler.donald/mac-ssd-not-booting-try-5-fixes-to-boot-up-your-mac-710d02aec288>)

18. Can't boot back to Mac OS from Windows : bootcamp - Reddit

(https://www.reddit.com/r/bootcamp/comments/bgwqvg/cant_boot_back_to_mac_os_from_windows/)

19. live environment - How do I get my Mac to boot from an Ubuntu USB key?

(<https://askubuntu.com/questions/28495/how-do-i-get-my-mac-to-boot-from-an-ubuntu-usb-key>)

20. My Mac won't start: How to fix the white screen issue?

(<https://blog.pcrisk.com/mac/12137-my-mac-wont-start-how-to-fix-white-screen>)

Step 2: URLs are listed in Step 1 under each result.

Result	Annotator1	Annotator2
1	match	match
2	match	match
3	match	match
4	not match	not match
5	match	match
6	match	match
7	not match	not match
8	match	match
9	match	match
10	match	match
11	match	match
12	match	not match
13	match	match
14	not match	not match
15	not match	not match
16	match	match
17	match	not match
18	not match	not match
19	not match	not match
20	not match	match

Step 3:

Cohen's kappa $\kappa = \frac{P(A)-P(E)}{1-P(E)}$

$$P(A) = \frac{17}{20} = 0.85$$

$$P(E) = \sum_c P(c|A1) * P(c|A2) = \frac{13}{20} * \frac{12}{20} + \frac{7}{20} * \frac{8}{20} = 0.53$$

$$\kappa = \frac{0.85-0.53}{1-0.53} \approx 0.68$$

Step 4:

Substantial Agreement. The inter annotator agreement is on the substantial high level, i.e. the two annotators agree with each other on most results. By results 12, 17 and 20, the two annotators annotate the results differently. It may be because the two annotators have different antecedents for the same query or they have different world knowledge.

For example, in query 12, both annotators' computers run on Mac OS Big Sur. Annotator 1 says it matches, because he knows that the reason that his computer doesn't boot is irrelevant to the version of operating system. But annotator 2 doesn't know this fact and only wants the result about Big Sur. Thus he considers that the result doesn't match.

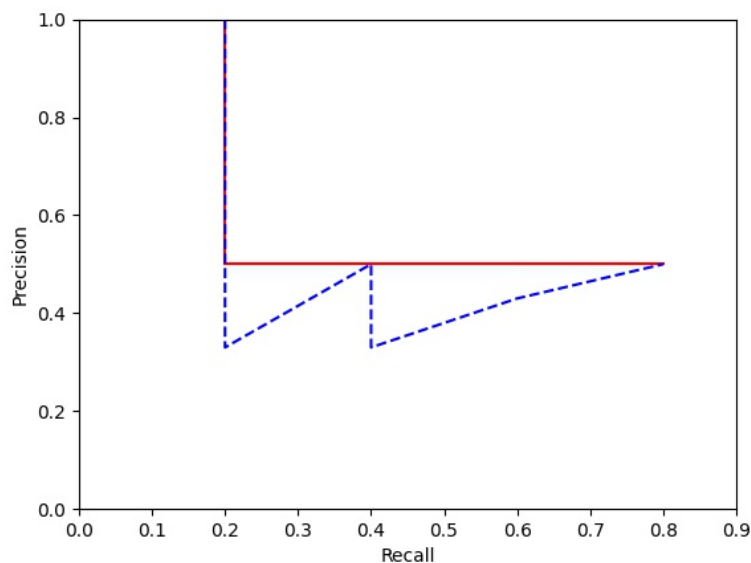
In query 17, Annotator 1 says it matches, because his Mac does have an SSD. Annotator 2 says it doesn't match, because his Mac has an HDD.

In query 20, annotator 1 says it doesn't match, because his computer won't boot, but doesn't have white screen(maybe blue screen). Annotator 2's computer won't boot and has white screen, so he considers that the result matches.

Task 3

- $Precision = \frac{TP_s}{TP_s + FP_s}$
- $Recall = \frac{TP_s}{TP_s + FN_s}$

k	Result Set	Precision	Recall
1	127	1.0	0.2
2	127, 9	0.5	0.2
3	127, 9, 10	0.33	0.2
4	127, 9, 10, 2	0.5	0.4
5	127, 9, 10, 2, 35	0.4	0.4
6	127, 9, 10, 2, 35, 32	0.33	0.4
7	127, 9, 10, 2, 35, 32, 41	0.43	0.6
8	127, 9, 10, 2, 35, 32, 41, 64	0.5	0.8



Task 4

An advantage of using IDF over stop word list is that you don't have a fixed set of words that maybe must be expanded in the future. Expanding a stop word list will change all the weights as a consequence. Due to the Zipf's law if a term is contained in all the documents is most likely to be a stop word. If the IDF value of a term is 0 (occurs in all documents) then will have the same effects as if it was included in the stop word lists.

Advantage of using stops words lists are that we could prevent the calculation of the IDF for stops and we can be more precise not including words that maybe occurs in every document but aren't stops.

Task 5

Document-at-a-time processing: Postings lists are ordered according to both docID and PageRank. The cosine similarity is computed based on this scheme. The query-document similarity score of each document is computed one by one, i.e. when the score of the current document is calculated, then calculate the score of the next one.

Term-at-a-time processing: No PageRank as weight. Process the postings list of a query term completely, then process the postings list of the next query term. Use an accumulator for each encountered docID.

Programming Task

Subtask 1

The ranking method implemented is based on tf-matching-score with log frequency weighting. Given a query the method **tf_matching_scores** will output a dictionary with the tf-matching-score for all relevant documents. A *relevant* document contains at least one term of the query at least one time.

In the implementation we iterate through the documents to calculate the log frequency weights for each term in the document ($W_{t,d}$) and then we calculate the tf-matching-score for each document. We use a log tf because the relevance does not increase proportionally with term frequency. After that we rank the results with the method **ranking_table** which ranks the results and print a ranking table to the terminal.

Code

```
1  #...
2
3  def tf_matching_scores(self, query: str) -> dict:
4      """Calculate tf-matching-score for all documents given the query.
5
6      Args:
7          query (str): Query against which we calculate the document's score
8
9      Returns:
10         dict: For each document the tf-matching-score
11         """
12     query = query.lower()
13     query = query.split()
14
15     docs_weights = {}
16     tf_matching_scores = {}
17
18     # retrieve the news title and text to calculate tf-weights
19     with open(self.filename, 'r') as file:
20         reader = csv.reader(file, delimiter = '\t')
```

```
21
22     #iterate through each row of the table
23     for row in reader:
24         # skip table header
25         if( row[0] == 'id' ): continue
26
27         #(doc_id , url , pub_date , title , news_text) = row
28         doc_ID = int(row[0])
29         news_title = row[-2]
30         news_text = row[-1]
31
32         docs_weights.update({ doc_ID: []})
33
34         tokenizer = nltk.RegexpTokenizer(r"\w+")
35         normalized_news_title = tokenizer.tokenize(news_title.lower())
36         normalized_news_text = tokenizer.tokenize(news_text.lower())
37
38         news_terms = normalized_news_title + normalized_news_text
39
40         for term in query:
41
42             if( term in news_terms ):
43                 log_tf = math.log( dict(Counter(news_terms))[term] )
44                 docs_weights[doc_ID].append(1 + log_tf)
45             else:
46                 docs_weights[doc_ID].append(0)
47
48         # delete all documents that are not relevant to the query
49         if( docs_weights[doc_ID] == [0,0] ):
50             docs_weights.pop(doc_ID)
51
52         #calculate tf-matching-scores
53         for doc_ID in docs_weights:
54             tf_matching_scores.update( {doc_ID: sum(docs_weights[doc_ID])} )
55
56     return tf_matching_scores
57
58 def ranking_table(self , scores: dict):
59     """Print a ranking table given a dictionary of scores
60
61     Args:
62         scores (dict): Datastructure that contains all scores
63     """
64     sorted_dict = sorted(scores.items(), key=itemgetter(1), reverse=True)
65     ranked_list = []
66     rank = 0
67     old_score = 0
68
69     for items in sorted_dict:
70         doc_ID = items[0]
71         score = items[-1]
72
73         # don't rank if the score is zero
74         if( score == 0 ): continue
75
76         if( old_score == score):
```



```
77         ranked_list.append((rank, doc.ID, score))
78     else:
79         rank += 1
80         ranked_list.append((rank, doc.ID, score))
81         old_score = score
82
83     # print ranking table
84     #print only top 10
85     print("Rank \t", "Doc \t", "Score")
86     print("-"*30)
87     counter = 0
88     for row in ranked_list:
89         if ( counter > 9 ): break
90         rank, doc_ID, score = row
91         print(rank, "\t", doc_ID, "\t", round(score, 5))
92         counter += 1
93
94
95
96 if __name__ == "__main__":
97     filename = 'assignment3/code/postillon.csv'
98     search = Search(filename=filename)
99
100    print("\n"*3)
101    print("1. Query: olympische sportbund")
102    scores = search.tf_matching_scores("olympische sportbund")
103    search.ranking_table(scores)
104
105    print("\n"*3)
106    print("2. Query: rot wein")
107    scores = search.tf_matching_scores("rot wein")
108    search.ranking_table(scores)
109
110    print("\n"*3)
111    print("3. Query: kinder sind faul")
112    scores = search.tf_matching_scores("kinder sind faul")
113    search.ranking_table(scores)
```

OUTPUT:

```
1      1. Query: olympische sportbund
2 Rank      Doc      Score
3 -----
4 1         4194     2.69315
5 2          276     2.60944
6 3         1632     2.09861
7 4          3925     2.0
8 4          4759     2.0
9 5          2018     1.69315
10 5          2146     1.69315
11 5          4990     1.69315
12 6           228     1.0
```

13	6	548	1.0
14			
15			
16			
17			
18	2. Query: rot wein		
19	Rank	Doc	Score
20	-----		
21	1	165	2.38629
22	2	1063	1.69315
23	2	1210	1.69315
24	2	1872	1.69315
25	2	2212	1.69315
26	2	2332	1.69315
27	2	2837	1.69315
28	2	2978	1.69315
29	2	3712	1.69315
30	3	150	1.0
31			
32			
33			
34			
35	3. Query: kinder sind faul		
36	Rank	Doc	Score
37	-----		
38	1	3763	5.4012
39	2	2789	5.07944
40	3	1827	4.89037
41	4	583	4.77259
42	5	4228	4.60944
43	6	1494	4.19722
44	6	2658	4.19722
45	7	2011	4.07944
46	8	150	3.94591
47	8	4887	3.94591