Information Retrieval (CS60092)
Mid Semester Examination, 23-09-2019, Autumn 2019-2020
CSE, IIT Kharagpur
Total Marks: 70

Name_____, **Roll No**_____
Instructions: Answer all the questions in the same order as given in the question paper.

1. **[2+2+4+2=10]**
   (a) Give two justifications or reasons to keep the document frequencies of the terms in the inverted index dictionaries. **[2]**

   > **Solution:** Query optimization in boolean retrieval, **[1]**
   > Query-Document relevance calculation in ranked retrieval **[1]**

   (b) Consider a set of N news articles (documents) out of which x articles contain the word "India" and y articles contain the word "Pakistan" (where $x, y \leq N$). Find out the best possible order of posting list merging times for the following queries given the inverted index for the dataset. **[2]**
      i. "India" AND NOT "Pakistan"

      > **Solution:** $O(x + y)$ **[1]**

      ii. "India" OR NOT "Pakistan"

      > **Solution:** $O(N)$ **[1]**

   (c) Are the following statements true or false? (Explain why?– You can give small examples.)
      i. In Boolean retrieval system, stemming never lowers precision. **[2]**

      > **Solution:** False. **[1]**
      > E.g., For query "automatic" it will retrieve documents containing words like "automate", "automation", etc. along with the truly relevant documents containing the word "automatic". **[1]**

      ii. Skip pointers help in faster merging of posting lists for all kinds of boolean queries. **[2]**

      > **Solution:** False. **[1]**
      > E.g., For queries like "x OR y", it is essential to visit every document-id in the posting list of either terms. Thus skip pointers will not help make the merging faster in such queries. **[1]**

   (d) Considering the below table suggest best possible query processing order for the following query. –"Economy" AND ("India" OR "Automotive") AND ("Ola" OR "Uber")– **[2]**

Term	Postings size
"Ola"	7,846
"Uber"	5,871
"Economy"	45,782,574
"Automotive"	12,548
"India"	5,68,974

Solution: 1. –("India" OR "Automotive")–, and –("Ola" OR "Uber")– separately. Let the resulting posting lists be L_1 and L_2.
2. –L_1 AND L_2–. Let the resulting posting list be L_3.
3. –L_3 AND "Economy"–. **[2 if all of above are correct]**

2. **[2+3+5=10]**

(a) Consider a permuterm index created for a dataset containing 4 documents. Each document contains 4 words each of 4 characters length. "od$fo" is one of the permuterms present in the index. Find out the maximum possible number of original vocabulary terms (present in the dataset) for the permuterm "od$fo". Give answer with explanation/derivation. **[2]**

Solution: 1 (only one). **[1]**
Irrespective of the exact permuterm, or the length and number of words and documents, each permuterm can represent only one actual vocabulary term. **[1]**

(b) Given two character strings s_1 and s_2, the edit distance between them is the minimum number of edit operations (allowed operations: insert a character, delete a character, replace a character by another) required to transform s_1 to s_2. Compute the edit distance between "Google" and "Yahoo". Write down the 6×5 matrix of distances between all prefixes. **[3]**

Solution: 6

		g	o	o	g	l	e
	0	1	2	3	4	5	6
y	1	1	2	3	4	5	6
a	2	2	2	3	4	5	6
h	3	3	3	3	4	5	6
o	4	4	3	3	4	5	6
o	5	5	4	3	4	5	6

[3 if everything is correct]

(c) Fun fact: Czechoslovakia was a sovereign state in Central Europe until its peaceful dissolution into the Czechia (Czech Republic) and Slovakia in 1993. Now consider a search engine query "Czechoslovakia". For this query, you want find out substitute queries. Clearly, "Czechia" and "Slovakia" seem to be very good choices. However, you are stuck with the question: which one of "Czechia" and "Slovakia" is a better substitute. Answer the following questions to find out.

i. Let's try bigram matches. List down the **bigrams** of query term and also the suggested substitutes. Consider terminal dollar sign ($) in all the cases. **[2]**

Solution: Bigrams
"Czechoslovakia": {cz, ze, ec, ch, ho, os, sl, lo, ov, va, ak, ki, ia, a$}
"Czechia": {cz, ze, ec, ch, hi, ia, a$}
"Slovakia": {sl, lo, ov, va, ak, ki, ia, a$} **[0.5 for each correct, and 2 if all correct]**

ii. Now using the bigrams sets, find out the Jaccard coefficient between each query and substitute pair. **[1]**

Solution: J("Czechoslovakia","Czechia")$=\frac{2}{5}$ **[0.5]**
J("Czechoslovakia","Slovakia")$=\frac{4}{7}$ **[0.5]**

iii. Based on your findings, which substitute is a better choice and why? **[1]**

Solution: "Slovakia" is a better choice due to higher Jaccard coefficient. Higher Jaccard coefficient means more similarity. **[1]**

iv. Instead of bigram or any other n-gram matches, what other metric could have been used for the same? (Limit your answers to metrics for term comparisions only.) **[1]**

Solution: Edit distance. **[1]**

3. **[1+4+5=10]**
Condsider the below system parameters.

Symbol	Parameter	Value
s	average seek time (before read/write)	4 ms $= 4 \times 10^{-3}$ s
b	transfer time per byte	0.03μs $= 3 \times 10^{-8}$ s
ν	processor's clock rate	10^9 s^{-1}
p	low level operations (compare or swap)	0.01μs $= 1 \times 10^{-8}$ s

Now imagine you have a dataset. You have already parsed the documents in the dataset and also saved the *(termID,docID)* pairs in the disk in blocks. Let the size of each *(termID,docID)* pair be 8 bytes and you have 100 Billion (100 Billion $= 10^{11}$) such pairs which are stored in 100 equal sized blocks in disk. Now you want to construct inverted index using Blocked Sort Based Indexing. Answer the following questions.

(a) What is the size of each block in bytes? **[1]**

Solution: Size of each block $= 10^9 \times 8$ bytes **[1]**

(b) First you read each of the blocks (one block at a time) into the memory, sort it and write it back to the disk block. Find the total time required for this. **[4]**

Solution: Time to seek and read each block into memory $= s+8\times10^9\times b = s+8\times10^9\times3\times10^{-8}$ $= 240.004$s
Time to sort each block $= (10^9 \log_2(10^9)) \times p = 298.973$s
Time to seek and write each block back into memory $=240.004$s
Total time for all the blocks $= 100 \times (240.004 + 298.973 + 240.004) = 77898.1\text{s} \equiv 21.64$ hours.

(c) Next you apply the following merging algorithm-1 to have a fully sorted list of *(termID,docID)* pairs. Find the total time required (Answer in terms of s, b, ν, p. No need of actual value). **[5]**

```
l=1;
while l < 100 do
    h = l + 1;
    X=Read(B_l);
    while h ≤ 100 do
        Y=Read(B_h);
        Merge the list of (termID,docID) pairs of X and Y;      // (2× list_size) comparisions
        Keep the first half of the merged list as X and the second half as Y in the memory;
        Write Y to B_h;
        h ← h + 1
    end
    Write X to B_l;
    l ← l + 1;
end
```

Algorithm 1: MERGE(B_1,$\cdots$,B_{100}). Input: pointers to blocks (B_1,$\cdots$,B_{100}).

Solution: Number of times B_1 is read = 1
Number of times B_2 is read = 2
Number of times B_i is read = i
Number of times B_{99} is read = 99
Number of times B_{100} is read = 99
Number of times B_1 is written = 1
Number of times B_2 is written = 2
Number of times B_i is written = i
Number of times B_{99} is written = 99
Number of times B_{100} is written = 99
Total number of block reads and writes = $2 \times (99 + \frac{99\times100}{2}) = 99 \times 102$
Total read/write time = $99 \times 102 \times (s + 8 \times 10^9 \times b)$
The total number of comparisions in merging a block pair of (X,Y) = 2× list_size = 2×10^9
Total number block pairs (X,Y) merged by the algorithm = $\sum_{l=1}^{99} \sum_{h=l+1}^{100} 1 = (100-1) + (100-2) + \cdots + 1 = \frac{99\times100}{2}$
Total time spent in merging = $\frac{99\times100}{2} \times 2 \times 10^9 \times p$
Total time = $99\times102\times(s+8\times10^9\times b) + \frac{99\times100}{2} \times 2\times10^9 \times p = 99\times102\times240.004 + 99\times1000 =$ 2522560.392s= 700.7 hours

4. **[2+2+2+2+2=10]**
Consider the postings list $< 100, 179, 234, 349, 405, 480, 523 >$. Memory units are addressed in bytes (8-bit blocks).

   (a) If fixed byte encoding is to be used to keep the postings (doc ids), what should be the size fixed for each posting and why? **[2]**

   Solution: 2 bytes. **[1]**
   523, the largest posting requires 2 bytes. **[1]**

   (b) Instead if we use varible byte encoding then find out how many bytes will be needed in total for the whole posting list? How much of space have we saved in comparison to the previous one? **[2]**

   Solution: 13 bytes. **[1]**
   1 byte saved, as in fixed we used 14 bytes. **[1]**

   (c) Instead of saving the postings, if we save first posting and then the gaps only (in fixed byte encoding), what should be the size fixed for each posting? In total how much of space will it require? **[2]**

   Solution: 1 byte. **[1]**
   7 bytes in total. **[1]**

   (d) Now, let's save the first posting and gaps list in variable byte encoding. In total how much of space will it require? How much of space have we saved in comparison to the previous one? **[2]**

   Solution: 7 bytes. **[1]**
   No space saved. **[1]**

   (e) Write the gamma encodings of the first posting and the last item in the gaps list. **[2]**

Solution: $(100)_{10} = (1111110100100)_{\gamma}$. **[1]**
$(43)_{10} = (11111001011)_{\gamma}$ **[1]**

5. **[5+2+3=10]**
   (a) Write if the following statements are true or false (no explanation needed).
      i. For a certain query term q, if document d_1 has a higher tf-idf score than document d_2, then the term q occurs more frequently in d_1 than d_2. **[1]**
      ii. Raw value of inverse document frequency of every vocabulary term is always finite. **[1]**
      iii. If the idf of term t_1 is higher than t_2 in a corpus, then the sum of term frequencies of t_1 over all the documents will also be higher than that of t_2. **[1]**
      iv. If the precision of a system is low, then the recall will be high. **[1]**
      v. If the precision@k for a query increases, then recall@k also increases. **[1]**

   Solution:
      i. True
      ii. True
      iii. False
      iv. False
      v. True

   (b) For a query q, the truly relevant documents are $\{d_1,d_4,d_5,d_6,d_9\}$. If a retrieval system gives ranked result as $\{d_{10},d_5,d_6,d_9,d_1,d_2,d_3,d_8,d_7,d_4\}$ for q, then calculate the MAP score of the system for the query set $Q = \{q\}$. **[3]**

   Solution: $\frac{1}{5}(\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{10}) \approx 0.643$

   (c) If all the queries in a system are one-term queries always, then do we still require idf or we can manage with just tf. Explain. **[2]**

   Solution: No, we don't require idf anymore. Because for single word queries, the idf remains the same for every term-doc pair thus becoming redundant.

6. **[3+4+3=10]**
   Consider the following corpus.
   d_1:"Amazon announces festival discounts on Fire product range in India."
   d_2:"Flipkart follows Amazon and announces bigger discounts."
   d_3:"Amazon rain forest is on fire."

   After removing stop-words {in, and, is, on}, and converting to lower case, answer the following questions. **Use logarithms with base 2**.
   (a) Write the raw term frequencies (in a matrix – terms in rows and documents in columns) and inverse document frequencies. **[3]**
   (b) Find out the tf-idf scores of each document (use **ltc** weighting) and write it in matrix (terms in rows and documents in columns). **[5]**
   (c) Find scores of each document for query "India fire" (use **nnn** weighting) based on cosine similarity scores. Which document is the most relevant one? **[2]**

Solution:

term	tf			idf	tf-idf (ltc)		
	d_1	d_2	d_3		d_1	d_2	d_3
amazon	1	1	1	0.00	0	0	0
announces	1	1	0	0.58	0.175	0.203	0
festival	1	0	0	1.58	0.477	0	0
discounts	1	1	0	0.58	0.175	0.203	0
fire	1	0	0	1.58	0.175	0	0.251
product	1	0	0	1.58	0.477	0	0
range	1	0	0	1.58	0.477	0	0
india	1	0	0	1.58	0.477	0	0
flipkart	0	1	0	1.58	0.0	0.552	0
follows	0	1	0	1.58	0.0	0.552	0
bigger	0	1	0	1.58	0.0	0.552	0
rain	0	0	1	1.58	0.0	0	0.684
forest	0	0	1	1.58	0.0	0	0.684

[3+4]

Scores for query "India fire"

	d_1	d_2	d_3
score	0.652	0	0.251

[2]

d_1 is the most relevant document to the given query. **[1]**

7. **[1+1+2+6=10]**
Consider a system with 100 documents in total. For a query q, there are exactly 5 relevant documents in the system. However, the system's retrieval algorithm provides the top-10 relevant documents for each query. Answer the following questions for query q.

   (a) Find the minimum and maximum precision possible? **[1]**

   Solution: Max=0.5 **[0.5]**
   Min=0 **[0.5]**

   (b) Find the minimum and maximum recall possible? **[1]**

   Solution: Max=1 **[0.5]**
   Min=0 **[0.5]**

   (c) Calculate the minimum and maximum MAP possible? Describe the cases when MAP will be maximum and minimum. **[2]**

   Solution: Max=0.822, when top-5 of the retrieved top-10 are the relevant ones. **[1]**
   Min=0, when none of the retrieved ones are relevant. **[1]**

   (d) Consider a dataset with documents as $\{d_1,d_2,d_3,d_4,d_5,d_6,d_7,d_8,d_9,d_{10}\}$. For a query q, the truly relevant documents are $\{d_1,d_4,d_5,d_6,d_9\}$. There is a retrieval system which randomly picks any five of the documents and gives it as result to each and every query. Now for this system, calculate the **expected values of** *Precision*@1, *Precision*@2, *Precision*@3, *Precision*@4, *Precision*@5 and MAP if the query is q.

Solution: E[*Precision*@1]=$\frac{{}^5C_1 \times 1}{{}^{10}C_1} = 0.5$ **[1]**

E[*Precision*@2]=$\frac{{}^5C_1 \times \frac{1}{2} +^5 C_2 \times 1}{{}^{10}C_2} \approx 0.28$ **[1]**

E[*Precision*@3]=$\frac{{}^5C_1 \times \frac{1}{3} +^5 C_2 \times \frac{2}{3} +^5 C_3 \times 1}{{}^{10}C_3} \approx 0.16$ **[1]**

E[*Precision*@4]=$\frac{{}^5C_1 \times \frac{1}{4} +^5 C_2 \times \frac{2}{4} +^5 C_3 \times \frac{3}{4} +^5 C_4 \times 1}{{}^{10}C_4} \approx 0.09$ **[1]**

E[*Precision*@5]=$\frac{{}^5C_1 \times \frac{1}{5} +^5 C_2 \times \frac{2}{5} +^5 C_3 \times \frac{3}{5} +^5 C_4 \times \frac{4}{5} +^5 C_5 \times 1}{{}^{10}C_5} \approx 0.06$ **[1]**

E[MAP]=$\frac{1}{5}\sum_{i=1}^{5} P(\text{relevant doc at } i) \times E(Precision@i \big| \text{relevant doc at } i)$
=$\frac{1}{5}\sum_{i=1}^{5} \frac{1}{2} \times E(Precision@i \big| \text{relevant doc at } i)$
=$\frac{1}{10}\sum_{i=1}^{5} E(Precision@i \big| \text{relevant doc at } i)$
=$\frac{1}{10}(1 + \frac{13}{18} + \frac{68}{3\times36} + \frac{196}{4\times84} + \frac{350}{5\times126}) \approx 0.3489$ **[1]**