Q1 Exam Instructions
0 Points

This exam is open book/note/internet, but you are expected to do the work on your own, without the help of other people (remote or local.) You are expected to complete the test in 50 minutes; it will cut you off after 60, but the extra 10 is for uploading handwritten answers; you are expected to complete your work in 50 minutes. Watch your time, and do not spend too long on any question. The times given are estimates - they are not enforced, but if you spend longer than the given time, you probably should go on and come back later.

You may ask questions in the office hours WebEx room during the hours it is staffed (hours sent via email): https://purdue.webex.com/purdue/j.php?MTID=m7433bffc48eafa61dcb8257f071b1790

Clarifications will be posted at: https://docs.google.com/document/d/1zchfWPWl1pY_5nCJQt4Zf-XZBNzKEOQAnLycrJsNeYc/edit?usp=sharing

Q1.1 Purdue Honor Code
0 Points

As a Boilermaker pursuing academic excellence, I pledge to be honest and true in all that I do. Accountable together - We are Purdue.

- ○ I agree with the Purdue Honor Code.
- ○ I do not agree with the Purdue Honor Code.

Q1.2 Answer upload method
0 Points

- (●) I will enter my answers in the text boxes and/or upload responses for each question. You must click "submit answer" before time expires. You can go back and change submitted answers (and resubmit) up until the time limit is reached, or you submit and view the completed exam.
- ( ) I will upload my entire answer set to Gradescope as a single PDF under "Midterm 1 (Paper only submission)"; answers submitted here will not be viewed or graded.

Q2 Inverted Index (6 minutes)
6 Points

Given the following inverted index:

	C:	F:	I:	L:	O:	R:
	Term: df	***DocID: tf***	***DocID: tf***	***...***		
5:	congressional: 4	D1: 1	D4: 1	D5: 1		
10:	district: 2	D1: 3	D4: 2	D5: 2	D6: 2	
15:	four: 2	D4: 1	D6: 3			
20:	house: 4	D4: 1	D5: 1	D6: 3		
25:	president: 16	D1: 3	D2: 8	D3: 2		
30:	representative: 8	D4: 2	D6: 2			
35:	senate: 2	D8: 2				

(Not all terms and documents are shown, but you won't need others to answer the question.)

Add the following document to the index (assume stemming and stopword removal has been done):
D9: "house representative rokita runs senate house".

You can simply show row/column numbers and new cell value where you make changes, you don't need to give the full

index. If you need to insert a row or column in between existing ones, just use a number in between the row/column numbers given (e.g, "3C: clifton: 5")

```
20C: 5          (one additional document contains house)
20O: D9: 2      (two occurrences of house in D9)
30C: 9
30L: D9: 1
35C: 3
35I: D9: 1
31C: rokita: 1  (adding new term in alphabetical order)
31F: D9: 1      (with the document for the new term)
32C: runs: 1
31F: D9: 1
```

Did you notice the flaw in the index? It didn't impact this question, but look at the document frequency for district. Then the number of documents containing district.

No files uploaded

(Side note - this is old news, former district four representative Rokita lost the Senate race, and is now running for Indiana Attorney General.)

Q3 Ad-hoc retrieval / index use (18 minutes)

10 Points

Given the following inverted index, which includes raw document frequency counts (number of documents containing the term) and the raw term count (number of times the term appears in the document) for each term/document pair. Assume there are 1024 documents total, and 10,000 distinct terms; you are seeing only a subset of the documents and terms:

	C:	F:	I:	L:	O:	R:
	Term: df	***DocID: tf***	***DocID: tf***	***...***		
5:	congressional: 4	D1: 1	D4: 1	D5: 1		
10:	district: 2	D1: 3	D4: 2	D5: 2	D6: 2	
15:	four: 2	D4: 1	D6: 3			
20:	house: 4	D4: 1	D5: 1	D6: 3		
25:	president: 16	D1: 3	D2: 8	D3: 2		
30:	representative: 8	D4: 2	D6: 2			
35:	senate: 2	D8: 2				

You are also given a table with normalized document lengths:

DocID	$\sqrt{\sum_{i \in terms} (tf_i * {}^{N}/_{df_i})^2}$	$\sqrt{\sum_{i \in terms} \left(\log_2(tf_i + 1) * \log_2 {}^{N}/_{df_i}\right)^2}$
D1	**1200**	**38**
D2	**900**	**36**
D3	**400**	**28**
D4	**1100**	**37**
D5	**1200**	**37**
D6	**1100**	**36**

A (2 points): Using a Boolean Retrieval model, give the returned documents for the query "district AND four AND (congressional OR representative)":

Only items D4 and D6 contain "4". They also contain "district", and both contain "representative", so the or clause is satisfied, so the answer is D4, D6

Note that if we used a ranked boolean, we might rank D4 higher, since both parts of the or clause are satisfied.

B (2 points): Compute the TF*IDF cosine similarity score between D4 and the query "congressional district four". Use

raw term counts and N/n for IDF:

sim(q,D4) = (1*1*1024/4 + 1*2*1024/2 + 1*1*1024/2) / (sqrt(1+1+1)*1100) approx 0.9
Note: Some calculated the denominator using the values in the index, this would be $\sqrt{((1024/4*1)^2+(1024/2*2)^2+(1024/2*1)^2+(1024/4*1}$ 1228, giving a TF*IDF score of approx 0.8; this was also acceptable. Likewise, using the count of documents (3, 4, 2) for document frequency was acceptable, this would give approx 0.6.

I didn't specify how to treat IDF for the query - if you used IDF to weight query terms, then the "1*" in the numerator would be replaced with the IDF, and the $\sqrt{(1+1+1)}$ with the square root of the sum if the squared IDFs. This was also acceptable, but you had to be consistent, using the same approach in both numerator and denominator.

Ignoring the $\sqrt{(3)}$ from the query, since this doesn't impact ranking, was acceptable if there was any indication you realized this wasn't actually cosine similarity (which should have been obvious, since it would be >1, and thus couldn't be a cosine.)

C (3 points): Rank all documents in the index by the raw term count, N/n TF*IDF score (as computed in part B.) Ignore documents where the score is 0. You need not calculate all TF*IDF scores, but if you do not, you should explain how you arrived at your rankings.

D6 > D4 > D1 > D5
Other documents are TF*IDF 0, since they contain none of the query terms. Note that for ranking we can ignore query term weighting (since it is 1, and the normalization $\sqrt{(3)}$ is the same for all.) D5 is clearly the lowest, since it's term counts are strictly dominated by D1 and D4, and

for D6 four 3*1024/2>congressional 1*1024/4, and the denominator is $\geq$ others. If we just look at the numerators, we can see
D1: 1*1024/4+3*1024/2 = 1792
D4: 1*1024/4+2*1024/2+1*1024/2 = 1792
D5: 1*1024/4+2*1024/2 = 1280
D6: 2*1024/2+3*1024/2 = 2560
Looking at the denominators, we can see that D6 is clearly the largest (and in fact has an impossible TF*IDF score - noting this makes up for a point missed elsewhere in this question.) Since the numerator for D1 is larger, D4 is next, then D1, then D5.

Note that if you worked out the TF*IDF for everything using the provided normalizations, you'd see a value >1 (I changed the D6 term counts, but didn't update the document length approximations.) If you saw this and noted what happened and why, you got a "free pass" that made up for one missed point elsewhere on this question.

D (2 points): Compute the TF*IDF cosine similarity score between D1 and the query "congressional district four", but this time use $\log_2(1+tf)$ for term frequency and $\log_2(N/n)$ for inverse document frequency.

$sim(q, D1) = \frac{((1*\log_2(1+1)=1)*(\log_2(1024/4)=8)+1*(\log_2(3+1)=2)*(\log_2(1024/2)=9)}{\sqrt{(1+1+1)*38)}} \approx 0.4$
(Calculating the denominator using the values listed in the first table gives a normalization of approximatly 33 and a TF*IDF score still approximately 0.4)

E (1 point): If you were to rank all documents using $\log_2(1+tf)$ for TF and $\log_2(N/n)$ for IDF, do you think the ranking would be the same as in part C? If yes, briefly explain why. If no, give one example of a pair of documents that would

change order, and briefly describe why.

This could result in a change, since this essentially reduces the impact of terms with a high IDF or TF. So, for example, D4 has a greater advantage from having all three terms vs. the value D6 gets from having more of a single term.

That said, in this case it doesn't actually make a difference (although it widens the relative gap between D4 and D1). Using the same approach as C, we see:
D1: 1*8+2*9
D4: 1*8+log_2(3)*9+1*9
D5: 1*8+log_2(3)*9
D6: 2*9+2*9
We can see this give no change. D6 is clearly the largest, D5 the smallest. Again, D4 > D1, but this time not only is the denominator for D4 smaller, but the numerator is larger.

In this question to get credit you either had to show you knew why it could change, or explicitly calculate that it didn't change. Simply stating that log preserves order isn't enough, as we aren't taking the log of the entire result (which would preserve order), but of the individual terms.

No files uploaded

Q4 Evaluating ad-hoc information retrieval (10 minutes)

6 Points

We run the query "congressional district four" using a TF*IDF and BM25 retrieval model, and get the following ranked lists and scores:

BM25	
DocID	*Score*
D22	12.0
D5	9.0
D13	8.0
D10	5.0
D18	4.0
D7	2.0
D4	0.8

TF*IDF	
DocID	*Score*
D4	0.8
D8	0.7
D18	0.6
D9	0.4
D11	0.3
D12	0.2
D22	0.1
D21	0.1

Based on manual inspection of the entire corpus, we have determined that the relevant documents are D4, D7, D8, D10, D12, D18, and D22.

A (4 points): Which retrieval model does a better job? Explain your answer.

Precision and recall alone don't tell us much, the recall is the same, and the precision of BM25 is better only because TF*IDF returns more documents. The important thing is ranking.

Since the "ground truth" is unranked, documents are simply relevant or not, all we can do is see how good they are at returning relevant documents. Comparing the two, with Relevant (R) and Non-Relevant (N) we see:
BM25: R N N R R R R
TF*IDF: R R R N N R R N
Looking at it this way, we see that the key differences is BM25 has non-relevant as 2 and 3, TF*IDF as 4 and 5, otherwise they are the same except for the extra document. So I'd say that TF*IDF is clearly giving a better ranking.

B (2 points): Calculate rank-based precision at 5 documents (precision@5) for both models. If you aren't familiar with

precision@5, partial credit for using another scoring measure for ranked lists.

Both documents have three relevant in the first five, so precision@5 for both is 3/5.

Note that if you tried to use the scores, you lost a point (but not a reduction below 0 for either part). It is almost never meaningful to compare the scores between different retrieval model.

No files uploaded

Q5 Probabilistic Retrieval (6 minutes)

3 Points

A (1 point): The retrieval status value (RSV) in the BIM and BM25 methods is based on an odds ratio rather than a probability. Why is it okay to use the odds ratio instead of calculating the probability a document is relevant to a query?

Odds ratio is the probability of being relevant divided by the probability of being non-relevant. Since we assume a document is one or the other, the odds ratio $\frac{P(R)}{(1-P(R))}$ gives the same ORDER as $P(R)$. If we increase $P(R)$, we can see that the odds ratio must go up as well (since numerator increase and denominator decreases.)

B (2 points): Give two reasons why we would prefer it to why we base the RSV on the odds ratio instead of calculating the probability a document is relevant to a query. (Not just "why it is okay" from part A, but why it is actually preferable.)

The big advantage is that some unknown values cancel out, allowing us to compute an odds ratio even when we don't know enough to compute the probability. In addition,

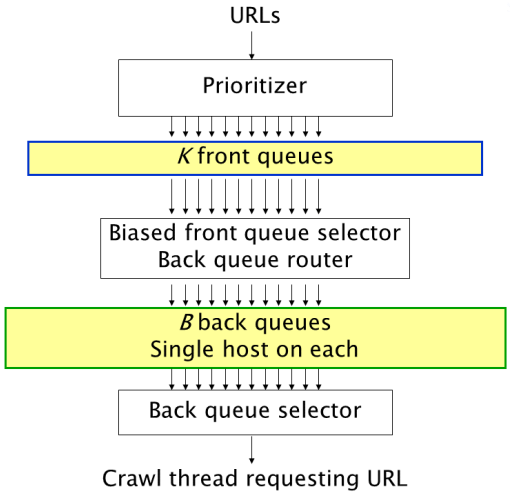
some calculable values can be factored out as constant per query, so we don't need to calculate them to get an order. Also, the probabilities tend to be very small, resulting in arithmetic precision errors, odds ratio (and better still, log odds ratio) avoid this problem.

No files uploaded

Q6 Politeness and Freshness (9 minutes)

6 Points

The Mercator method uses two sets of queues (front and back queues) to manage both age/freshness and politeness. All queues are FIFO.



We assign documents to be crawled to different front queues to ensure freshness, documents we want to crawl sooner are assigned to higher priority queues.

A (4 points): Given document A that on average changes once per day and was last crawled three days ago, and document B that changes on average once every six days and was last crawled 7 days ago, should A or B be assigned to a higher priority queue if we want to minimize age of documents? Show how you determine this.

There are a multiple ways we can figure age, but in any

case it is integrating the probability of a change at any given time over the elapsed time since the last crawl. You could either use any reasonable approach for the probability provided you capture the ratio that A is six times as likely to change as B at a given time; I will use the Poisson distribution
$\int_0^t \lambda e^{-\lambda x}(t-x)dx = \frac{\lambda t + e^{-\lambda t} - 1}{\lambda}$
(λ = 1 for A, 1/6 for B). The integral should be over 0-3 for A, 0-7 for B.

A: $\int_0^3 1e^{-1x}(t-x)dx = 3 + e^{-3} - 1 \approx 2.0$
B: $\int_0^7 \frac{1}{6}e^{-(1/6)x}(t-x)dx = \frac{1/6 \cdot 7 + e^{-(1/6)7} - 1}{1/6} \approx 2.9$

Since we expect B to be more out-of-date, we assign it to a higher priority queue.

Note that intuitively it would seem that A should be more out of date; the Poisson distribution assumes a change is more likely early with long tails balancing out the average, which is why B actually works out to have a higher expected age. Using an assumption of a flatter distribution with the same average could well result in A having a higher expected age. For full credit, I wanted to see some idea of integrating over time rather than just intuition.

B (2 points): Suppose we have a host that we want to make a request to at most once per second. What would be a simple way to enforce this politeness criterion in the Mercator method?

Each host falls only in a single back queue. If we store the "last accessed" time in each back queue, and only allow pulling from that queue after a second has elapsed, we will not hit that host more than once per second.

Note that we want the timer to be per queue, not overall. Otherwise we are limiting ourselves to 86,400 pages

crawled/day, which isn't going to index much of the web.

No files uploaded

Q7 Final question (nothing to answer, just a question)

0 Points

Why all the questions about Congressional district four? If you are registered to vote on campus, you should know the answer.

Remember to save/submit your exam!

Midterm 1

● **UNGRADED**

5 DAYS, 4 HOURS LATE

STUDENT
Unknown Student (removed from roster?)

TOTAL POINTS
- / 31 pts

QUESTION 1

Exam Instructions		0 pts
1.1	Purdue Honor Code	0 pts
1.2	Answer upload method	0 pts

QUESTION 2

Inverted Index (6 minutes)		6 pts

QUESTION 3

Ad-hoc retrieval / index use (18 minutes) 10 pts

QUESTION 4

Evaluating ad-hoc information retrieval (10 minutes) 6 pts

QUESTION 5

Probabilistic Retrieval (6 minutes) 3 pts

QUESTION 6

Politeness and Freshness (9 minutes) 6 pts

QUESTION 7

Final question (nothing to answer, just a question) 0 pts