**National University of Computer & Emerging Sciences, Karachi**
**Spring-2023 FAST School of Computing**
**Final Exam (Sol)**
**May 29, 2022, 08:30 AM – 11:30 AM**

| **Course Code:** CS4051 | **Course Name:** Information Retrieval |
|---|---|
| **Instructor Name:** Muhammad Rafi | |
| **Student Roll No:** | **Section:** |

Instructions:

- Return the question paper with the answer book.
- Read each question completely before answering it. There are **7 questions** on **4 pages**.
- Start each question on a separate staring page. Answer all queries in a question –using new line for each. Give code segment, algorithm, diagram where it is necessary.
- In case of any ambiguity, you may make assumptions. But your assumptions should not contradict with any statement in the question paper.

**Time: 180 minutes** **Max: 100 Marks**

| **Basic IR + Posting list + Dictionary + Tolerant** | | |
|---|---|---|
| **Question No. 1** | **<CLO#1>** | **[Time: 20 Min] [Marks: 10]** |

a. Give a list of all possible bi-grams for the following list of strings, there is no need to attached any sentinel character. The result set of bi-gram should be in sorted order.
S= {ask, mask, flask, task, bask}

Bi-grams ={as, fl, ha, la, ma, sk, ta}

b. What are the components of a Posting List?

Postings List is a data structures often used in information retrieval systems, which records that a term (text feature) appeared in a document (at a specific position as well). There are two vital components of a posting list: (i) Dictionary or Lexicon or Vocabulary list and (ii) postings/position entries.

c. Define what is the syntax of a general wildcard query. What is the best data structure to process these queries with-out any false positive output?

A general wildcard query in IR is of the form **mo*s*er** where more than one astrisk appears in the query string. Positional index is the best data structures to answer these queries without any false positive in the result set.

d. How Heap's Law and Zipfs Law help in determining the size of a large inverted index for a given collection? Explain with an example.

Regardless of the values of the parameters for a particular collection, Heaps' law suggests that (i) the dictionary size continues to increase with more documents in the collection, rather than a maximum vocabulary size being reached, and (ii) the size of the dictionary is quite large for large collections. Zipf's law states that given a large sample of words used, the frequency of any word is inversely proportional to its rank in the frequency table. Both of these laws help in estimating the size of a posting list of an arbitrary collection.

| **Indexing + Vector Space Model** | | |
|---|---|---|
| **Question No. 2** | **<CLO#1>** | **[Time: 20 Min] [Marks: 10]** |

a. What are some of the drawbacks of Vector Space Model (VSM)? [5]

b. Consider the following document collection:

D1: w1 w2 w3 w1
D2: w2 w4 w5
D3: w1 w2 w6 w3

Using the vector space model, find the possible ranking of all the given documents for the query **Q: w4 w5**, You can use tf-idf(t, d) = tf(t, d) * idf(t) and idf(t) = log [ n / df(t) ] + 1 in your calculations. [5]

Consider the following table constructed for VSM from the given data:

| | D1 | D2 | D3 | Q | IDF | D1:TF*IDF | D2:TF*IDF | D3:TF*IDF | Q: TF*IDF |
|---|---|---|---|---|---|---|---|---|---|
| ***w1*** | 2 | 0 | 1 | 0 | 2 | 3.169925 | 0 | 1.584963 | 0 |
| ***w2*** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| ***w3*** | 1 | 0 | 1 | 0 | 2 | 1.5849625 | 0 | 1.584963 | 0 |
| ***w4*** | 0 | 1 | 0 | 1 | 3 | 0 | 2.584963 | 0 | 2.584963 |
| ***w5*** | 0 | 1 | 0 | 1 | 3 | 0 | 2.584963 | 0 | 2.584963 |
| ***w6*** | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 2.584963 | 0 |

Now, the best ranking of VSM comes from Cosine Similarity.

Hence,

Cos(D1,Q)= 0

Cos(D2,Q)=13.312

Cos(D3,Q)= 0

Ranking order will be (D2, D1, D3) or (D2 D3 and D1)

| IR Evaluation and Relevance Feedback | | |
|---|---|---|
| **Question No. 3** | **<CLO#1>** | **[Time: 20 Min] [Marks: 10]** |

a. The following list of Rs and Ns represents relevant (R) and non-relevant (N) returned documents in a ranked list of 12 documents retrieved in response to a query from a collection of 1000 documents. The top of the ranked list (the document the system thinks is most likely to be relevant) is on the left of the list. This list shows 5 relevant documents. Assume that there are 8 relevant documents in total in the collection. [1.5X4]

R R N N R N N N R N N R

i. What is precision of the system on the top 12?

From the given information, we can see that tp=5; fp=7 and fn=7-5=2 so for precision we have Precision = tp / (tp+fp) = 5/12 = 0.416

ii. What is F1 of the system on the top 12?

Let's find recall for F1: we know Recall = tp/ (fn+tp) = 5/(2+5)=0.714 hence
F1= 2 X (Precision * Recall) / (Precision + Recall)
F1= (2X0.416X0.714) / (0.416+0.714) = 0.594 / 1.13 = 0.525

iii. What is the largest possible MAP that this system could have?

The maximum MAP possible when the remaining three relevant documents retrieved next to these 12 documents.
MAP = 1/8 * { 1/1+2/2+3/5+4/9+5/12+6/13+7/14+8/15} = 0.619

iv. What is the smallest possible MAP that this system could have?

The minimum MAP possible when the remaining three relevant documents found as the last documents from the collection.

MAP = MAP = 1/8 * { 1/1+2/2+3/5+4/9+5/12+6/998+7/999+8/1000} = 0.435

b. Omar has implemented a relevance feedback web search system, where he is going to do relevance feedback based only on words in the title text returned for a page (for efficiency). The user is going to rank 3 results. The first user, Uzma, queries for:

Query: banana slug

The top three titles returned are:

d1: banana slug Ariolimax columbianus
d2: Santa Cruz mountains banana slug
d3: Santa Cruz Campus Mascot

Uzma judges the first two documents relevant, and the third non-relevant. Assume that Omar's search engine uses term frequency but no length normalization nor IDF. Assume that he is using the Rocchio relevance feedback mechanism, with $\alpha = \beta = \gamma = 1$. Show the final revised query that would be run. (Please list the vector elements in alphabetical order.) [5]

Consider the following table:

| | Q | d1 | d2 | d3 |
|---|---|---|---|---|
| ***ariolimax*** | 0 | 1 | 0 | 0 |
| ***banana*** | 1 | 1 | 1 | 0 |
| ***campus*** | 0 | 0 | 0 | 1 |
| ***columbains*** | 0 | 1 | 0 | 0 |
| ***cruz*** | 0 | 0 | 1 | 1 |
| ***mascot*** | 0 | 0 | 0 | 1 |
| ***mountains*** | 0 | 0 | 1 | 0 |
| ***santa*** | 0 | 0 | 1 | 1 |
| ***slug*** | 1 | 1 | 1 | 0 |

Using the equation given below:

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

Q(opt) = <1/2, 2, 0, 1/2, 0, 0, 1/2, 0, 2>

| Probabilistic IR, Language Model and Neural IR | | |
|---|---|---|
| **Question No. 4** | **<CLO#2>** | **[Time: 30 Min] [Marks: 20]** |

a. Consider the following document collection:

D1: w1 w2 w3 w1
D2: w2 w4 w5
D3: w1 w2 w6 w3

Using the uni-gram language model, find the possible ranking of all the given documents for the query Q: w4 w5, you can use the following formula for the probability calculation $p(w_i) = (count(w_i) + 0.5) / (dN + 1)$. [5]

| | w1 | w2 | w3 | w4 | w5 | w6 |
|---|---|---|---|---|---|---|
| D1 | 0.625 | 0.375 | 0.375 | 0.125 | 0.125 | 0.125 |
| D2 | 0.125 | 0.375 | 0.125 | 0.375 | 0.375 | 0.125 |
| D3 | 0.375 | 0.375 | 0.375 | 0.125 | 0.125 | 0.375 |

P(Q/M(D1)) = 0.125 * 0.125 = 0.0156
P(Q/M(D2)) = 0.375 * 0.375 = 0.1406
P(Q/M(D3)) = 0.125 * 0.125 = 0.0156
Hence the ranking will be (D2, D3 and D1) or (D2, D1 and D3)

b. Give two similarities and two difference between Skip Gram (SkipGram) and Continuous Bag of Word(CBOW) Models. [5]

CBOW is trained to predict a single word from a fixed window size of context words, whereas Skip-gram does the opposite, and tries to predict several context words from a single input word. According to the original paper, Mikolov et al., it is found that Skip-Gram works well with small datasets, and can better represent less frequent words. However, CBOW is found to train faster than Skip-Gram, and can better represent more frequent words.

c. Consider the following word-embedding matrix

| | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 |
|---|---|---|---|---|---|---|---|---|---|---|
| W1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| W2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| W3 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| W4 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| W5 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

Explain how can you find a pair of most similar words based on the learned embedding. Perform all necessary calculations and heuristic to achieve the same. [10]

The most common heuristic that can be used for finding similarity is to get the most common dimension of any pair of word in the embedding space and perform cosine similarity between the vector of these words.

Based on this heuristic we get the two pair of words (W1, W3) and (W2, W4)

Now, for cosine of these vectors we get

Similarity (W1, W3) = 0.80

Similarity (W2, W4) = 0.89

Hence the most similar words can be found in O(N*d).

From the paper of Basker Mitra

| Clustering + Classification | | |
|---|---|---|
| **Question No. 5** | **<CLO#3>** | **[Time: 30 Min] [Marks: 20]** |

Consider the following collection of documents:

D1: w1 w2 w3 w1
D2: w2 w4 w5
D3: w1 w2 w6 w2
D4: w1 w2 w3

Assuming D1, D3 and D4 are relevant to a query and D2 is non-relevant.

Test document for classification D5: w2 w4 w6

a. Using feature vectors as term frequency and the k-Nearest Neighbors (KNN) with k=3 identify the class of test instance D5? [5]

Dictionary Order = < w1, w2, w3, w4, w5, w6>
D1= <2,1,1,0,0,0> D2=<0,1,0,1,1,0> D3=<1,2,0,0,0,1> D4=<1,1,1,0,0,0>
D5=<0,1,0,1, 0,1>
distance(D1,D5) =2.6457; distance(D2,D5) = 1.414; m(D3D5) = 1.732; m(D4D5) = 2
3- nearest neighbors are D2, D3, and D4 **hence D5 is relevant**.

b. Using the same feature vectors and the Rocchio's algorithm, classify the test instance D5? [5]

Consider the relevant and on-relevant documents centroid mass, we get

μ(Relevant) = 1/3 { D1+D3+D4} = <4/3,4/3,2/3,0,0.1>
and μ(Non-Relevant) =D2 =<0,1,0,1,1,0>

Now measuring distance for the two centroid masses,
distance(D5, μ(Relevant) ) = 5/3
distance(D5, μ(Non-Relevant) ) = 2
**D5 is Non-Relevant.**

c. Using k-means algorithm perform clustering of the given 4 documents. Initially, you can use k=2 with seeds = {D2 and D1}- Perform only 2 iterations of the algorithm. [5]

Using seeds D1= <2,1,1,0,0,0> D2=<0,1,0,1,1,0>

For 1st Iteration:

distance (D1, D3) =2 and distance (D2, D3) = 2.23
distance (D1, D4) =1 and distance (D2, D4) = 2

Both D3 and D4 belong to D1 and D2 is the other cluster. Hence the centorids of the two clusters would be μ(C1) =< 4/3,4/3, 2/3, 0,0,1/3> and μ(C2) =<0,1,0,1,1,0>

For 2nd Iteration:

distance (D1, μ(C1)) =0.88 and distance (D2, μ(C1)) = 2.1
distance (D3, μ(C1)) =1.20 and distance (D4, μ(C1)) = 0.66

distance (D1, μ(C2)) =2.64 and distance (D2, μ(C2)) = 0
distance (D3, μ(C2)) =2.23 and distance (D4, μ(C2)) = 2

document D1, D3 and D4 belong to μ(C1) and D2 belong to μ(C2)

d. Can K-means algorithm ever converge to have an empty cluster for any value of K greater than 2? How we can handle this situation? [5]

Yes. K-means can coverage to empty cluster as output. Empty clusters can be obtained if no points are allocated to a cluster during the assignment step. If you get an empty cluster, it has no center of mass. You can simply ignore this cluster (set k=k-1 for next iteration), or repeat the k-means run from a new initialization.

| Web Search + Web Crawling | | |
|---|---|---|
| **Question No. 6** | **<CLO#3>** | **[Time: 20 Min] [Marks: 10]** |

a. Illustrate the difference between a directory style web search(yahoo) and free text search (google). [5]

| Directory Search (Yahoo) | Text Search (Google) |
|---|---|
| - Yahoo style, directory search maintains a large hierarchical directory of terms, a user need to follow the topic based hierarchical path to get the information. | - Google style, simple text based search, autonomously crawl and index pages on text terms. The query terms are process through index and relevant pages are returned as result. |
| - It is very challenging to maintain such large directory | - Indexing is easy as compared to directory maintenance |
| - Relevant results from exploration of the hierarchical path. | - Relevant results as per the retrieval approach (pagerank). |

b. Outline at least 5 challenges that a web scale information crawler may come across. [5]

The web-sites generally block access through automatic crawlers, the crawler must be polite in accessing these websites. There can be several issues with physical access, network access and application layer of the web-host, a crawler must be robust to all these problem. Crawler should be distributed and scalable as it need to access millions of pages per unit time.

| Link Analysis | | |
|---|---|---|
| **Question No. 7** | **<CLO#2>** | **[Time: 40 Min] [Marks: 20]** |

a. Outline three differences between HITS and PageRank algorithms for link analysis. [5]

| **HITS** | **PageRank** |
|---|---|
| It gives two scores Hub and Authority for each page. | It gives one score per page. |
| It is executed at query time. | It is precomputed at indexing time. |
| It is query dependent. | It is independent from query. |
| It is not robust against web/link spams | It is robust against web-spams |
| Never favors pages, but can be manipulated for higher scores. | It favours old pages. It can also be manipulated. |

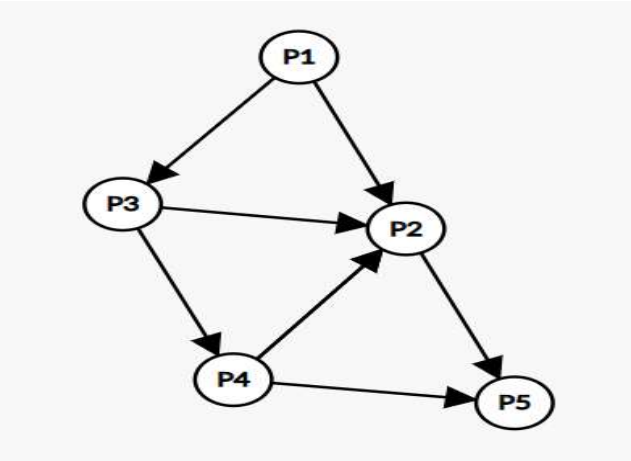
b. What are some of the interesting facts discovered while running HITS algorithm on a variety of queries? Give at least two. [5]

Running HITS across a variety of queries reveals some interesting insights about link analysis.

- Frequently, the documents that emerge as top hubs and authorities include languages other than the language of the query. This ensure cross language information retrieval.
- HITS is more stable for link analysis, a small change to link topology should not lead to significant changes in the ranked list of results for a query.

c. Consider a segment of web which only contains five pages. Assume that P1 connected to P2 and P3; P3 connected to P2 and P4; P2 connected to P5; and P4 connected to P2 and P5; (where connected means out-link to the page-directed arc). In a stepwise fashion, first provide an adjacency matrix for the given graph, later provide two iteration of HITS algorithm to find Authority Weight Vector ($A^2$) and Hub Weight Vector ($H^2$), You can use Euclidian normalization to the resultant vectors in each step. [10]

Consider the following graph as per the given information:




Now, Consider the probability metric

P =

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

Let A be the connectivity matrix for the given graph from part a. We know that $h^1 = P \cdot a^0$ and $a^1 = P^T \cdot h^0$ , similarly $h^2 = P \cdot a^1$ and $a^2 = P^T \cdot h^1$

P=

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

$a^0$

| |
|---|
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |

$P^T$ =

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |

$h^0$

| |
|---|
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |

After multiplying and normalizing we get

| $a^1$ | |
|---|---|
| | 0 |
| | 0.832 |
| | 0.277 |
| | 0.277 |
| | 0.554 |

| $h^1$ | |
|---|---|
| | 0.554 |
| | 0.277 |
| | 0.554 |
| | 0.554 |
| | 0 |

| $a^2$ | |
|---|---|
| | 0 |
| | 0.453 |
| | 0 |
| | 0.226 |
| | 0.902 |

| $h^2$ | |
|---|---|
| | 0.645 |
| | 0 |
| | 0.645 |
| | 0.215 |
| | 0 |

**<The End>**