National University of Computer & Emerging Sciences, Karachi
Spring-2020 CS-Department
Midterm I
February 24, 2020 (Time: 11AM – 12Noon)

Course Code: CS317	**Course Name: Information Retrieval**
Instructor Name:	**Dr. Muhammad Rafi**
Student Roll No:	**Section No:**

- Return the question paper.
- Read each question completely before answering it. There are **3 questions and 2 pages.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict with any statement in the question paper.
- All the answers must be solved according to the sequence given in the question paper.
- Be specific, to the point and illustrate with diagram/code where necessary.

Time: 60 minutes. **Max Marks**: 40 points

Question No. 1 **[Time: 25 Min] [Marks: 20]**

Answer the following questions briefly using 4-5 lines of answer book. Be precise, accurate and to the point, only answer genuine query in the question. Each question is of 2 marks.

a. Define the data structures "Term Document Matrix"? What are its limitations?

A term-document matrix (Data Structures) is an important representation for text analytics. Each row of the matrix is a document vector, with one column for every term in the entire corpus. Generally, there are large number of terms and reasonable small number of documents in comparison of terms.
There are two critical limitations of this data structures for text analytics: (1) Sparsity – large number of entries are zeros. (2) very rigid in nature cannot be linearly scaled.

b. Define Inverted Index? What are its components?

An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. In simple words, it is a hashmap like data structure that directs you from a word to a document or a web page. It has two components: (1) Dictionary or lexicon or vocabulary - which contains all the distinct terms from the collection. (2) Posting - documents ID in which these terms appear.

c. What are some of the problems associated with Stemming in IR?

Stemming algorithms are typically rule-based. You can view them as heuristic process that sort-of lops off the ends of words. A word is looked at and run through a series of conditionals that determine how to cut it down. There are two problems (1) Over-stemming comes from when too much of a word is cut off. This can result in nonsensical stems, where all the meaning of the word is lost or muddled. Or it can result in words being resolved to the same stems, even though they probably should not be. (2) Under-stemming is the opposite issue. It comes from when we have several words that actually are forms of one another. It would be nice for them to all resolve to the same stem, but unfortunately, they do not.

d. For a conjunctive query of the form (t1 AND t2 AND t3), is processing postings lists in increasing order of their lengths (posting size) guaranteed to be optimal execution order? Explain why it is, or give an example where it isn't.

Processing postings list in order of size (i.e. the shortest postings list first) is usually a good approach. But it is not optimal e. g. in a conjunctive query with three terms:

t1 = 1,25,33 --> posting size 3
t2 = 2,33,44,75 --> posting size 4
t3 = 10,31,20,60,48. --> posting size 5

As we can see there is no document containing all three query terms. If we would have checked the first posting of the third list right at the beginning, we would have noticed that there is no intersection between the first and the third postings list. That would make any further search superfluous. Going in size order that is for t1 and t2 first would cost more.

e. What do we mean by Phrase Query? What is the best data structures for indexing, to answer these queries without false positive results?

Consider the query "Cross Language Information retrieval" it is a general phrase query. The intent of the user is to get the documents that contains the complete list of words in the same order. Positional Index can be used to answer this type of query without any false positive in the result-sets.

f. What is the idea behind a permuterm index for Information retrieval?

The idea behind Permuterm Index is to rotate wildcard query such that * goes to the end. Thus, you convert vague query to comparable query. For example, consider someone want to search for X* where X is a prefix string. In this kind of index every word is appended with a special character say $ at the end. Now every rotation (character-wise from left to right) of every word is kept as a primary index to get the corresponding word from the dictionary. In our example X* we will search for $X* in our permuterm index, you can think that * is uncertain but in $X* X is certain, hence the primary index mapped all string staring with prefix X.

g. Give an example of a term (text-feature) that falsely matches the wildcard query **mon*h** if the search were to simply use a conjunction of bigrams.

Consider the given query mon*h if the system uses bigrams index to retrieve documents, this query will be translated to "$m AND mo AND on AND h$" hence a document contains "moonish" qualify for it, which is a false positive.

h. What do we mean by post-processing in IR? What is it complexity?

The post processing in IR is a specialized phase of result-set processing. In some model of IR there is a chance of getting false positive documents against a given query due to the processing approach. The post processing tries to filter all the false positive documents. The complexity of this phase is directly proportional to number of documents in the result-set.

i. How can we estimate the numbers(distribution) of terms in a collection for indexing?

In information retrieval system, we often need to estimate the distribution of terms in a collection, we only know the size of collection as a whole. Heaps Law help us in determining how many terms will be there in a collection. It states that the vocabulary size can be estimated as a function of collection size $M = kT^b$ where k is a constant and b=0.4 as an exponent.

j. Why term-frequency –inverse document frequency (tf*idf) is a good weighting scheme for Vector Space Model (VSM)?

TF-IDF, which stands for term frequency—inverse document frequency, is a scoring measure widely used in information retrieval. TF-IDF is intended to reflect how relevant a term is in a given document. It has two components (tf) and (idf). The tf component ensure that a frequent term in a document get a higher score in the weighting scheme, while if this terms appears in more commonly in collection (other document), it would not have any discriminating criteria hence the score should be lesser for such term. The idf component provide this scoring function. In other words, tf-idf score for a term t in document d get a score as:

(1) highest when term occurs many times within a small number of documents (thus lending high discriminating power to those documents);
(2) lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
(3) lowest when the term occurs in virtually all documents.

Question No. 2	**[Time: 15 Min] [Marks: 10]**

a. What is meant by tolerant retrieval? Give an example of the situation in which a user has a vague information need. [5]

- Tolerant retrieval means the information retrieval system deliver you best possible answer on the submission of query by compensating the error in the query both syntax and semantics and guide the users for effectively retrieving information process through information retrieval system.
- Examples of tolerant retrieval from web searches are:
1. Spelling suggestion for query word
2. Wild card support
3. Context sensitive information retrieval
4. Search options based on statistical results of the systems.

A user may not be aware of the correct spelling of the term and run search on "procrastination" the search engine suggests the option as "Showing results for procrastination or Search instead for procrastination"

b. How the search engines provide spelling correction as a part of user experience? Explain with an example? What are the underlying principles on which general spelling correction approaches are built? How their functionality is used for user engagement? [5]

The search engine has become a primary means to accessing information on the web, However, the user may not be sure about the spelling of certain terms hence their query is ambiguous. The results are incorrect or inconclusive. Whenever, search engine found that the query term is not returning a reasonable number of relevant documents, it tries it best to come up with more relevant query term with the help of submitted query.

They do suggest certain spelling correction and hence gain more trust of the users. There are two broad categories of spelling mistakes that search engine does (1) Isolated work correction – in which single word or term from the query is considered for correction. A dictionary is used to rectify with the closest term utilizing any measures or yardstick (2) Context sensitive spelling corrections- in which they considered the entire query text as a guide to resolve the mistakes in terms. Hence able to correct longer sentences (query text). The user found it very helpful and more commonly depends on these kind of supports with strong interaction with the systems.

Question No. 3 **[Time: 15 Min] [Marks: 10]**

Consider the following small collection of documents from a collection and a query Q:

Doc 1: w1 w2 w5 w1
Doc 2: w4 w1 w3 w6
Doc 3: w2 w3 w1 w4
Q: w2 w3 w4

a. Compute and present the documents and query vectors for the above, when the idf scores for the terms w1, w2, w3, w4, w5 and w6 are 0.25, 0.01,0.05,0.1,0.35, and 0.35 respectively. Use the well-known term frequency and inverse document frequency weighting scheme (tf*idf) as vector weights. [5]

	idf	tf-d1	tf-d2	tf-d3	tf-q	tf*idf -d1	tf*idf -d2	tf*idf -d3	tf*idf -q
w1	0.25	2	1	1	0	0.5	0.25	0.25	0
w2	0.01	1	0	1	1	0.01	0	0.01	0.01
w3	0.05	0	1	1	1	0	0.05	0.05	0.05
w4	0.1	0	1	1	1	0	0.1	0.1	0.1
w5	0.35	1	0	0	0	0.35	0	0	0
w6	0.35	0	1	0	0	0	0.35	0	0

Required vectors:
d1: <0.5,0.01,0,0,0.35,0>
d2: <0.25,0,0.05,0.1,0,0.35>
d3: <0.25,0.01,0.05,0.1,0,0>
q: <0,0.01,0.05,0.1,0,0>

b. Using the cosine of angle between the query and document vector, list the documents in decreasing order of relevance? [5]

	Cos(d1,q)	Cos(d2,q)	Cos(d3,q)
w1	0	0	0
w2	0.0001	0	0.0001
w3	0	0.0025	0.0025
w4	0	0.01	0.01
w5	0	0	0
w6	0	0	0
	0.0001	0.0125	0.0126

Ranking will be D3, D2, and D1

<Best of luck for your exam>