National University of Computer & Emerging Sciences, Karachi
Spring-2018 CS-Department
Midterm 1 (Sol)
27th February 2017, 10:30AM – 11:30AM

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 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. Illustrate at least three problems while performing "Tokenization" of documents.

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. It is very challenging aspect of information retrieval, the tokenization process need to decide about a lot of different aspect of a natural language/processing like:

1. Direction of parsing for tokenization.
2. Should it treat space as separator for token e.g. Les Vegas a single or two tokens.
3. Treatment of punctuation characters like hyphen (-) co-ordinated.
4. If there is no space in between word boundary, how it will decide about tokens like in Japanese or German compound nouns.

b. Are there some of the assumptions of Boolean Retrieval Model for IR?

There are two very fundamental assumptions for Boolean Model

1. User knows the document; he is looking for. He knows the features present/absent in a document.
2. Documents are easily machine readable and all such features are easy to extract.

c. Explain the proximity operator (/k) in query of the form “w1 w2 /k”

The proximity query of the form " x y /k". Where x and y are terms from the collection and k is an integer. The result-set contains documents where the two separately matching terms x and y occurrences are within a specified distance say k, where distance is the number of intermediate words.

d. Why permuterm index is efficient for a wildcard query with only one "*" symbol?

Consider the wildcard query with single "*" like: m*n. The key is to rotate such a wildcard query so that the * symbol appears at the end of the string – thus the rotated wildcard query becomes n$m*. Next, we look up this string in the permuterm index, where seeking n$m* (via a search tree) leads to rotations of (among others) the terms man and moron. Now that the permuterm index enables us to identify the original vocabulary terms matching a wildcard query, we look up these terms in the standard inverted index to retrieve matching documents.

e. A bigram index is used to retrieve document for wildcard query "te*ti*al". Suggest how a Boolean query on a bigram index would look like for this? Give an example of term that may contain in the result-set.

The Boolean query using bi-grams will be "$t AND te AND ti AND al AND l$"
Example is: testimonial

f. What is the limitation of isolated term correction approach to spelling correction?

In isolated-term correction, we attempt to correct each term independent of the actual number of query terms. For example, query like "flew form Heathrow" would be try to correct each term "flew" "form" and "Heathrow". This type of correction is applicable to all lexical terms. For example, a single query term "carot" can easily be corrected to "carrot". There are two techniques for this method (i) Edit distance and (ii) k-gram overlap.

g. How Block-Sort Based Indexing(BSBI) gain advantages over traditional inverted indexing approach?

The traditional inverted indexing technique uses termID-docID pairs from each document, and try to sort their termID–docID pairs in memory to construct a dictionary. The posting list for all such term may be large and can be kept into secondary memory. This can be very expensive for large collection. Block Sort Based Indexing, first try to segments the collection into parts of equal size, second sorts the termID-docID pairs of each part in memory, these two steps generally fits into memory as we can divide the collection into our preferred size. The sorted intermediate results from each such part immediately stored into continuous blocks on disk (sequential disk reads are faster). When the complete collection is processed, it then merges all these blocks into final inverted index, processed and stored on disk.

h. How can we estimate the 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.

i. Is there any relation between Heaps Law and Zipfs Law? Explain

Zipfs law leads to heaps law. The zipfs law concerning the frequencies on individual wrods(token) in a collection. It state that, if t1 is the most common term in the collection, t2 is the next most common, and so on, then the collection frequency cfi of the ith most common term is proportional to 1/i. Mathematically, cfi directly proportion to 1/i. Under mild assumptions, the Heaps law is asymptotically equivalent to Zipf's law concerning the frequencies of individual words within a text. 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 = kTb where k is a constant and b=0.4 as an exponent. This is a consequence of the fact that the type-token relation (in general) of a homogenous text can be derived from the distribution of its types.

j. Compare the difference between Block-Sort Based Indexing (BSBI) and Single-Pass In Memory Indexing (SPIMI).

Block-Sort Based Indexing (BSBI)	**Single-Pass In Memory Indexing (SPIMI)**
- BSBI uses continuous disk space to collect all terms from document collections by dividing collection into equal parts, iteratively. - It uses a data structures to collect termID and docID into memory. - The running time is proportional to (T log T) where T is Number of terms in the collection. Dominated by sorting of terms in a collection.	- SPIMI add posting directly to posting list and small posting list are stored into the continuous disk blocks. - There is no need to map termID and docID pairs and hence no sorting is required. Faster and efficient. - The running time is linear in term of T(number of terms in the collection). - SPIMI also support compression of posting lists.

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

The Boolean retrieval model has several drawbacks, one is the query formulation, which is not straight forward. Below are some of the information needs from user's, given in plain English, you need to transform these into Boolean Model queries. Using the term statistics from the collection. Suggest a best order for your queries processing, using your knowledge on inverted index and related concepts.

Some of the term statistics from the dictionary / inverted indexes are given:

Benefit	167	Hot	319	Reduce	980
Cholesterol	6430	Information	23100	Seeds	12323
Cold	767	Local	23900	Success	23451
Eat	37623	Oats	12398	Swim	32414
Effective	231653	Optimize	879	Technique	387912
Exam	21345	Prepare	1213	Water	1998721
Flax	3124	Rate	982345		
Global	21342				

Q1: Eating flax seeds is more effective at reducing cholesterol than eating oats.

Eat AND Flax AND Seeds AND Effective AND Reduce AND Cholesterol AND Oats
Posting sizes: (37623) ∩ (3124) ∩ (12323) ∩ (231653) ∩ (980) ∩ (6430) ∩ (12398)
The most effective query processing order will be:
Reduce AND Flax AND Cholesterol AND Seeds AND Oats AND Eat AND Effective

Q2: Swimming benefits in hot water or cold water

(Swim AND Benefit AND Water) AND (Hot OR Cold)
Posting sizes: (32414) ∩ (167) ∩ (1998721) ∩ (319 U 767)
The most effective query processing order will be:
Benefits AND (Hot OR Cold) AND Swim AND Water

Q3: success rate of exam preparation techniques

Success AND Rate AND Exam AND Prepare AND Technique
Posting sizes: (23451) ∩ (982345) ∩ (21345) ∩ 1213 ∩ 387912
The most effective query processing order will be:
Prepare AND Exam AND Success AND Technique AND Rate

Q4: local or global optimization techniques

(Local OR global) AND Optimize AND Technique
Posting sizes: (23900 U 21342) ∩ 879 ∩ 387912
The most effective query processing order will be:
Optimize AND (Local OR Global) AND Technique

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

Consider the following documents in a collection.

Doc 1: new home sales top forecasts
Doc 2: home sales rise in july
Doc 3: increase in home sales in july
Doc 4: july new home sales rise

a. Draw the inverted index that would be built for this collection. Assuming "in" is the only stop word that you can filter. Give both the dictionary and posting list for each term. [8]

Inverted Index

Dictionary	**Posting Lists**
forecast	forecast->1
home	home->1->2->3->4
increase	increase->3
july	july->2->3
new	new->1->4
rise	rise->2->4
sale	sale->1->2->3->4
top	top->1

b. Assume someone is applying Lemmatization, what terms(lexeme) the words [] = {sales, forecasts, rise} are mapped to? [2]
With Lemmatization these terms mapped to the following:

Sales → Sale
Forecasts → Forcast
Rise → Rise