National University of Computer & Emerging Sciences, Karachi
Spring-2019 CS-Department
Midterm I (Sol)
February 27, 2019; 9AM – 10AM

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. What is an Inverted Index? What are its components?

Inverted index is a kind of index on text which is composed of vocabulary and a list of occurrences in the documents. It has two main components: (i) dictionary – a collection of all the terms appears in the collection and (ii) posting list – a list of document ids which contains the term.

b. Outline at least three differences between Stemming and Lemmatization.

Stemming	**Lemmatization**
It is a heuristic- rule based approach, generally fast and use a single term. It generates unreadable tokens. Stemming algorithms err on the side of being too aggressive, sacrificing precision in order to increase recall.	It is a rigor process that uses a dictionary and uses context to determine the lemma, considered a slow approach. It generates readable lexeme from the dictionary. Lemmatization offers better precision than stemming, but at the expense of recall.

c. What do we mean by Ad hoc Information retrieval?

Ad hoc Information retrieval is a kind of standard retrieval task. In which a user specifies his arbitrary information need through a one-off, user initiated query (a free text query) which initiates a search (executed by the information system) for documents which are likely to be relevant to the user.

d. 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.

e. How Single-Pass In-memory Indexing (SPIMI) gain advantages over Block-Sort Based Indexing (BSBI) approach?

SPIMI adds a posting directly to its postings list. Instead of first collecting all termID-docID pairs and then sorting them like BSBI, each postings list is dynamic (i.e., its size is adjusted as it grows) and it is immediately available to collect postings. This has two advantages: It is faster because there is no sorting required, and it saves memory because we keep track of the term a postings list belongs to, so the termIDs of postings need not be stored. As a result, the blocks that individual calls of SPIMI-INVERT can process are much larger and the index construction process as a whole is more efficient hence these are some of the advantages over BSBI.

f. Suggest at least two optimizations for holding (in memory) dictionary from an inverted index to facilitate retrieval in the setup of an offline static collection.

In an inverted index there are two components (i) dictionary and (ii) posting lists. The dictionary is kept in memory as to facilitate the quick search on the terms. It is always desirable to have a compact and compressed dictionary. The offline static collection, we can run a lot of optimization. The terms can be kept with dynamic strings as to save over fixed length term string. Hence a handful of memory can be saved. One more optimization is to keep all the terms from the dictionary in a form of a large string. With every term index kept the pointer to the start of the string, the next term pointer determines the end of the previous term. Hence a very compact representation is possible.

g. What do we mean by Trailing Wildcard Queries? What is the best data structure for the index, for serving these queries?

A query such as mon* is known as a trailing wildcard query, because the * symbol occurs only once, at the end of the search string. A search tree on the dictionary is a convenient way of handling trailing wildcard queries: we walk down the tree following the symbols m, o and n in turn, at which point we can enumerate the set |W| of terms in the dictionary with the prefix mon. Finally, we use |W| lookups on the standard inverted index to retrieve all documents containing any term in W.

h. What do we mean by permuterm index? Give permuterm vocabulary for “gaga”

In a permuterm index each dictionary entry is index by placing a $ character at the end, the string so formed is rotated single character at a time to produce all permuterm vocabulary. For example, gaga will be like gaga$, aga$g, gaga, agag, $gaga
Permuterm index is very helpful in supporting wildcard queries with single *.

i. 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.

j. 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 helps 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.

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

a. What do we mean 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. What is a multi-term query? How query correction and suggestion work for such a query? Explain by an example. [5]

A multi-term query is a query comprises of multiple terms. The free text queries are general form of multi-term queries. The query e.g. ***ticket from cheap air ticket from London to New York***, is a multi-term query. The query correction for such a query is quite challenging as individual terms may be correct but a contextual information is not correct for example the same query may be incorrectly written as ***ticket form cheap air ticket from London to New York*** hence the determining a correct form of the query requires the understanding of the entire sentence. The statistical methods for context sensitive correction is used in this case.

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

Consider the following documents in a collection.

Doc 1: multi label classification text
Doc 2: text classification in python
Doc 3: label of text is classification
Doc 4: classification of label in text

a. Draw the inverted index that would be built for this collection. Assuming "in" and "of" are the stop words that you can filter. Give both the dictionary and posting list for each term. [5]

Inverted Index

Dictionary	**Posting Lists**
classification	classification->1,2,3,4
is	is -> 2
label	label ->1,3,4
multi	multi ->1
python	python ->2
text	text -> 1,2,3,4

b. How this inverted index can be used to process the query "text AND label AND python", How this query can be optimized? [5]

The query "text AND label AND python" if it is processed in the given order the cardinality of each term will be |term| = 4, |label|=3, and |python|=1 hence the first pair of term intersect and there will be three attempt of matching document Ids. Later the result set will be intersecting with posting list of |python| no document qualifies. If we run the query like: python AND label AND text least effort is required.