COMP 336 Information Retrieval Midterm Examination, Spring 2000 March 20, 2000

Name: _____ Student ID: _____

1. Using the ***basic KMP algorithm***, determine the number of character positions to shift in the shift[] array for the pattern **`tomatoes`** [10]

	shift
t	1
o	1
m	2
a	3
t	4
o	4
e	4
s	7

Explain ***briefly*** how the algorithm can be improved by examining the pattern character which causes a mismatch with the text. Show the new values of the shift array when the improvement is applied. [10]

If after shifting the pattern, the new pattern character to be compared with the text string is the same as the pattern character that caused the mismatch, we can conclude that a match is not possible and thus shift the pattern further by a number of position as determined by the shift table entry for the new pattern character.

	shift
t	1
o	1
m	2
a	3
t	5
o	5
e	4
s	7

Give two ***major*** differences between the KMP and BM algorithms. [6]

Any two of the following (although the first two characterize BM better):

1) comparison is from right to left
2) the text character causing a mismatch is using to determine the shift (delta 1)
3) repeated substrings in the pattern are obtained from right to left to determine the shift (delta 2)

Explain how we can use KMP pattern matching to support prefix, suffix or infix truncation in the pattern. [9]

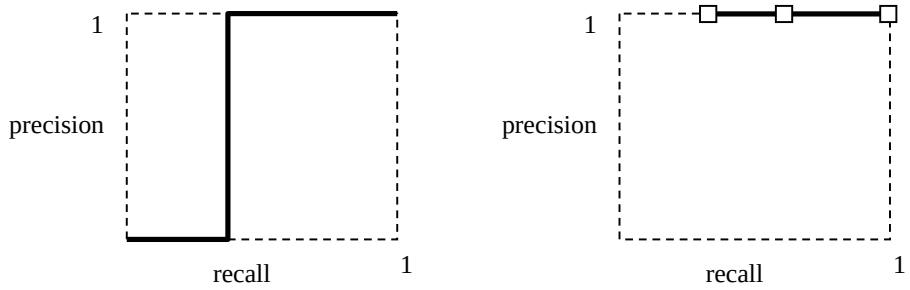
Prefix and suffix: nothing need to be done; e.g., *xyz and xyz* are equivalent to searching for xyz
infix: ijk*xyz is equivalent to searching for ijk and ***upon a match search onwards for xyz***

[morale of this question is that infix, prefix, and suffix truncation can be easily handled in full text scanning compared to an inverted file]

2. Fill in the precision, recall and fallout rates in the following table. It is assumed that there are a total of 100 documents, and there are only 5 relevant documents, which are marked with a √ in the first column. [30]

	Rank	doc ID	Recall	Precision	Fallout Rate
	1	1001	0	0	1/95
√	2	2873	1/5	1/2	1/95
√	3	3916	2/5	2/3	1/95
	4	0983	2/5	2/4	2/95
	5	8310	2/5	2/5	3/95
√	6	7892	3/5	3/6	3/95
	7	4562	3/5	3/7	4/95
√	8	4921	4/5	4/8	4/95
√	9	7934	1	5/9	4/95
	10	9248	1	5/10	5/95
...	. . .	...	...	...	...
	100	3861	1	5/100	95/95

Is the following precision/recall graph possible? Justify your answer. [5]



precision = number of relevant document retrieved / number of documents retrieved
recall = number of relevant retrieved / number of relevant documents in collection

We can see that when precision = 0, recall must be zero. Note that the diagram to the right is possible but interpolation would not produce the graph on the left.

3. The following inverted file shows the values of the postings list for three terms. In each postings entry, the first value is the document ID, and the second value is the term frequency of the term in the document. Notice that the inverted file contains other terms and documents which are not shown in the diagram.

stock	→	2, 10	9, 9			
. . .		. . .	. . .			
taiwan	→	9, 5	10, 3			
. . .		. . .	. . .			
value	→	2, 12	8, 1	9, 7	8, 1	9, 9

Given the query "taiwan stock", where the query term weights for "taiwan" and "stock" are both equal to 1, calculate the document scores of each document using tfxidf term weighting without tf or idf normalizaton, and the inner product similarity function. There are totally 100 documents in the collection. [15]

Doc 8 = Doc 18 = Doc 19 = 0 (not containing any query term)
Doc 2 = 10 log 100/2 = 10 log 50
Doc 9 = (5 + 9) log 100/2 = 14 log 50
Doc 10 = 3 log 100/2 = 3 log 50

If I want to normalize tf using tfmax normalization, can the invert file allow it? If yes, how? If no, what other information has to be kept in order to allow it? [7]

Definition: tfmax of a document is the largest term frequency within the document.

The answer is no, since an exhaustive search of the entire inverted file is required to obtain tfmax of a document. Some sort of "forward index" mapping a document ID to its tfmax is needed.

The answer "yes" is accepted as partially correct, if your answer says an exhaustive search is needed.

The answer "yes" is accepted as completely correct, if your answer says an exhaustive search is needed but it is impractical/difficult to do.

If I want to use cosine similarity measure to compute document scores, can the invert file allow it? If yes, how? If no, what other information has to be kept in order to allow it? [8]

Cosine similarity is the inner product normalized by the query and document lengths. The inner product can be computed based on tfxidf. Query weight is known (either by assuming them to be one, as given in the question, or using tfxidf weighting on the query). What is remaining is the normalization factor. Query vector length can be obtained without problem. However, finding the document length requires us to know the weight of ***every*** term in a document. It is impractical to obtain from the inverted file. Again, some sort of forward index is needed to map a document ID to its document length (or to a list of words it contained, from which the document length can be computed).