COMP 336 Information Retrieval Quiz 1, Spring 2002 **March 7, 2002**
Time Allowed: 35 minutes

Name: _____ **Student ID:** _____

[Note: Answer all questions in the space provided. Answers must be precise and to the point.]

1. [25] In the case study which evaluated the precision and recall of STAIR, it was found that precision is only 20% and recall is only 80% even after iterations of search, what might be the reason for the search not being able to reach 100% precision and recall? [You could give your own reasons that don't have to match the ones I gave in class, but justify your reasons.]

   Precision and recall are low because it is hard to formulate a complex Boolean query to represent what the user wants and that the same concept can be expressed in different ways in the documents.

   In legal cases, lawyers tend to look for specific information, e.g., car stolen in a public car park. If the query is "car and stolen" then it will return document talking about car stolen anywhere (e.g., from home), which is not exactly what the query means. That cause some retrieval documents are irrelevant to the query. It can be accounted for the low precision.

   On the other hand, if the query includes "car park" in the query, then there may be documents in which "car park" is inferred from situation (e.g., "After arriving the shopping, he parked his car … but later find that it was stolen") but not explicitly mentioned in the documents. These documents may be lost. Therefore, Boolean queries are difficult to write. That can be accounted for that the recall can't reach 100%.

   Finally, different ways of expression may cause problems, e.g., "my car was lost in the car park", but this is also a problem with vector space model (in fact, all systems based on *syntactic* keyword matching will have this problem).

   [Students: You may notice that the grading is very flexible. It is set to provoke your thoughts.]

2. [25] When an inverted file is used for answering Boolean queries such as A AND B AND C, it involves three steps:

	Change in time consumed when the number of documents increases
a) Searching the index	Increase slowly (vocabulary won't increase forever, and searching the index is a log (n) process). If you make correct assumption that the original documents have contained enough index terms and the new document won't increase the number of index terms, it can also be (i) not change.
b) Retrieving from disk the postings lists	Linear (because when the number of documents is double the lengths of the postings lists will on average double also)
c) Intersecting the lists	Linear (because merging/intersecting two sorted lists only requires examining the lists once)

   When the number of documents increases, indicate in the second column of the above table whether the time taken for the step will: (i) not change, (ii) increase slowly (i.e., sub-linearly), (iii) increase linearly, or (iv) increase exponentially with the number of documents indexed, assuming that the postings lists are sorted by document Ids. Briefly justify your choice below.

[See above table.]

3. [20] Explain why insertion is more expensive than retrieval. How much more expensive is the insertion of a document of *n* unique terms when compared to a query searching for one term?

   Given a document of *n* unique terms, you need to search the inverted file to locate the *n* index terms, read *n* postings lists, update the lists, and write them back to the inverted files for inserting that document. While searching a query for one term only involves locating one index term and reading a posting lists.

   Since compared with the access and writing of posting lists on disk, the time of searching the index terms can be omitted. Again assume that reading a postings list needs one disk I/O and write a postings list back to the inverted file needs one disk I/O too.

   Cost of insertion of a document of n unique terms = 2 disk I/O * n
   Cost of a query searching for one terms = 1 disk I/O

   Thus insertion of a document of n unique terms is 2*n* expensive of a query searching for one term.

4. [20%] Suppose there are only 5 unique terms in the collection, which contains a total of 100 documents. The term's term frequencies in a document *D* and their document frequencies are given below:

   $tf_{D,1} = 2$ $df_1 = 10$
   $tf_{D,2} = 0$ $df_2 = 20$
   $tf_{D,3} = 1$ $df_3 = 30$
   $tf_{D,4} = 5$ $df_4 = 20$
   $tf_{D,5} = 2$ $df_5 = 10$

   Write down the document vector when tf/tf_{max} * idf is used.
   $idf_1=\log_2(100/10)$ $w_{D,1}=2/5*idf_1=1.329$
   $idf_2=\log_2(100/20)$ $w_{D,2}=0/5*idf_1=0$
   $idf_3=\log_2(100/30)$ $w_{D,3}=1/5*idf_3=0.347$
   $idf_4=\log_2(100/20)$ $w_{D,4}=5/5*idf_4=2.322$
   $idf_5=\log_2(100/10)$ $w_{D,5}=2/5*idf_5=1.329$

D=<1.329, 0, 0.347, 2.322, 1.329>

Given the query vector, $Q = \langle 0, 1, 1, 0, 1 \rangle$, compute the inner product and cosine similarity values between *Q* and *D*.

Inner product: Sim (Q,D) = 0.347*1+1.329*1=1.676

Cosine Similarity: Cossim(Q, D)= $\frac{1.676}{\sqrt{(1^2+1^2+1^2)}*(1.329^2+0.347^2+2.322^2+1.329^2)}$ =0.322

5. [10] Which of the statements is true about the extended Boolean model:

   [**T**/F] Users cannot assign weights to query terms (e.g., A AND B, with A more important B)

   [T/**F**] For A OR B, the similarity score of a document is the same as the distance between the point representing the document and point $\langle 1,1 \rangle$.