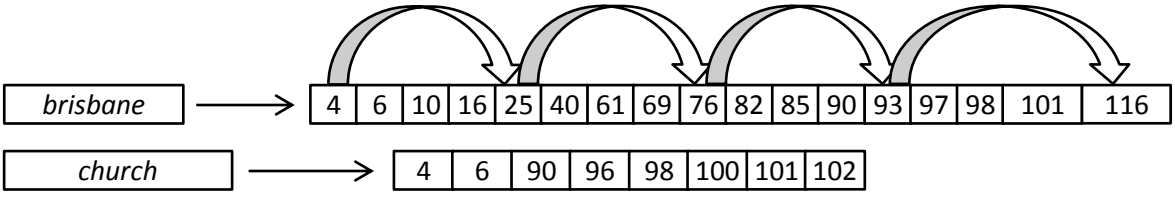
Information Retrieval (CS60092)
Mid-semester examination, Autumn 2012 – 2013

Time: 2 hours, **Full Marks:** 50

Attempt all questions.
Use of calculator is allowed.
State any assumptions made clearly.

Q. 1> Consider the Boolean query *brisbane* AND *church*. The postings lists of the two words are



shown. The list for *brisbane* is augmented with skip pointers.

(i) How many comparisons are required if neither of the lists had skip pointers? List the comparisons sequentially.

Ans. No. of comparisons: 19

Comparisons [<*a, b*> denotes *a* and *b* are doc-ids from *brisbane* and *church* lists respectively]: <4, 4>; <6, 6>; <10, 90>; <16, 90>; <25, 90>; <40, 90>; <61, 90>; <69, 90>; <76, 90>; <82, 90>; <85, 90>; <90, 90>; <93, 96>; <97, 96>; <97, 98>; <98, 98>; <101, 100>; <101, 101>; <116, 102>

(ii) How many comparisons are required in the situation shown, when *brisbane* has skips? List the comparisons sequentially.

Ans. No. of comparisons: 19

Comparisons: <4, 4>; <25, 6> (Skip? Check – Fail); <6, 6>; <10, 90>; <16, 90>; <25, 90>; <76, 90> (Skip? Check – Success); <93, 90> (Skip? Check – Fail); <82, 90>; <85, 90>; <90, 90>; <93, 96>; <116, 96> (Skip? Check – Fail); <97, 96>; <97, 98>; <98, 98>; <101, 100>; <101, 101>; <116, 102>

(iii) How often is a skip pointer used in the above situation?

Ans. Once [25 -> 76]

(iv) What is the trade-off issue faced involving time and space complexities when choosing the number of skip pointers for a given postings list?

Ans. More skips means shorter skip spans, and that we are more likely to skip. But it also means lots of comparisons to skip pointers, and lots of space storing skip pointers. Fewer skips means few pointer comparisons, but then long skip spans which means that there will be fewer opportunities to skip.

(v) What is a common heuristic for deciding the number of skip pointers, given that the size of the postings list is *P*?

Ans. $\sqrt{P}$ evenly-spaced skip pointers.

(vi) Are skip pointers useful for processing Boolean OR queries? Justify.

Ans. No, because skip pointers are useful only when intersecting two lists. In OR queries, we only need to find their union. **(3 + 3 + 1 + 1 + 1 + 1 = 10)**

Q. 2> List two cases of IR where achieving perfect recall is more important than high precision. **(2)**

Ans. a. Medical document search [Medical IR]

b. Legal/patent document search [Legal/Patent IR]

Q. 3> (i) What is the Levenshtein distance between: (a) *house* and *musket* (b) *basket* and *stables*?

Ans. (a) 4

[**Explanation (need not be provided): Option 1:** *house -> mouse -> muse -> muske -> musket*

Option 2: *house -> houset -> housket -> mouseket -> musket*

Other alternatives may exist]

(b) 5

Option 1: *basket -> sasket -> stasket -> stabket -> stablet -> stables*

(ii) What is the time complexity of the dynamic programming approach that is commonly used to calculate the Levenshtein distance between two strings of length $|s_1|$ and $|s_2|$?

Ans. $O(|s_1| \times |s_2|)$

(iii) What is the Jaccard coefficient between: (i) *idea* and *sidearm*? (ii) *boat* and *aboard*? Assume alphabet bigrams as the unit and take into account terminal dollar ($) for the computations (a word *cat* is assumed to be denoted as *cat$*). **(2 + 1 + 2 = 5)**

Ans. (i) *idea* is denoted as *idea$*; *sidearm* is denoted as *sidearm$*

Bigram set in term 1, *A = {id, de, ea, a$}*

Bigram set in term 2, *B = {si, id, de, ea, ar, rm, m$}*

Reqd. Jaccard Coefficient = $|A \cap B| / |A \cup B| = 3/8 = 0.375$

(ii) boat -> boat$; aboard -> aboard$

A = {*bo*, *oa*, *at*, *t$*}

B = *{ab, bo, oa, ar, rd, d$}*

Reqd. Jaccard Coefficient = 2/8 = 0.25

Q. 4> Consider the query *catholic church brisbane*. Assume the vector space model and the TF-IDF weighting scheme. The corpus consists only of the following documents:

d_1: *roman catholic church brisbane roman*

d_2: *church brisbane church church*

d_3: *catholic catholic protestant protestant all all brisbane*

d_4: *catholic church welcome catholic church brisbane*

Assign vector indices 1 – 7 to the vocabulary words in the following order:

roman, catholic, church, brisbane, protestant, all, welcome

(i) Now, assuming this order, clearly write the complete weight vectors for the query and each document. State the formula used for IDF.

Ans. IDF of a term *t*, $idf_t = \log_{10}(N/df_t)$ [Log base can be any other number as well, here we assume 10]

where N is the total no. of documents in the corpus; df_t (document frequency of *t*) is the no. of documents that *t* occurs in]

The tf-idf vectors are the reqd. weight vectors ($weight_{t,d} = tf_{t,d} \times idf_t$).

Vector	*roman*	*catholic*	*church*	*brisbane*	*protestant*	*all*	*welcome*
df	1	3	3	4	1	1	1
idf	0.602060	0.124939	0.124939	0.000000	0.602060	0.602060	0.602060
Query tf	0	1	1	1	0	0	0
Query tf-idf	0.000000	0.124939	0.124939	0.000000	0.000000	0.000000	0.000000
d_1 tf	2	1	1	1	0	0	0
d_1 tf-idf	1.204120	0.124939	0.124939	0.000000	0.000000	0.000000	0.000000
d_2 tf	0	0	3	1	0	0	0
d_2 tf-idf	0.000000	0.000000	0.374816	0.000000	0.000000	0.000000	0.000000
d_3 tf	0	2	0	1	2	2	0
d_3 tf-idf	0.000000	0.249877	0.000000	0.000000	1.204120	1.204120	0.000000
d_4 tf	0	2	2	1	0	0	1
d_4 tf-idf	0.000000	0.249877	0.249877	0.000000	0.000000	0.000000	0.602060

(ii) What are the lower and upper bounds for IDF of a term in a corpus according to the stated formula?

Ans. Lower bound = 0 [term present in all documents]

Upper bound = $\log_{10}(N)$ [term present in one document]

(iii) Rank the documents w.r.t. the query assuming the overlap score measure [**Hint:** The score for a query-document pair is the sum of the weights of the query terms in the document vector].

Ans. $Overlap\ Score(q,d) = \sum_{t \in q} tf - idf_{t,d}$

Thus, Overlap Score(q, d_1) = 0.124939 + 0.124939 + 0 = 0.249878

Overlap Score(q, d_2) = 0 + 0.374816 + 0 = 0.374816

Overlap Score(q, d_3) = 0.249877 + 0 + 0 = 0.249877

Overlap Score(q, d_4) = 0.249877 + 0.249877 + 0 = 0.499754

Thus, Reqd. Ranking: 1: d_4; 2: d_2, 3: d_1, d_3

(iv) Rank the documents assuming the cosine similarity as a scoring function. Show all steps of the computation. **(3 + 1 + 2 + 4 = 10)**

Ans.

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}$$

$$sim(q, d_1) = \frac{(0.125 * 0.125) + (0.125 * 0.125)}{\sqrt{(0.125^2 + 0.125^2)}.\sqrt{1.204^2 + 0.125^2 + 0.125^2}} = \frac{0.031}{0.177 * 1.217} = 0.144$$

$$sim(q, d_2) = \frac{(0.125 * 0.375)}{\sqrt{(0.125^2 + 0.125^2)}.\sqrt{0.375^2}} = \frac{0.047}{0.177 * 0.375} = 0.708$$

$$sim(q, d_3) = \frac{(0.125 * 0.250)}{\sqrt{(0.125^2 + 0.125^2)}.\sqrt{0.250^2 + 1.204^2 + 1.204^2}} = \frac{0.031}{0.177 * 1.721} = 0.102$$

$$sim(q, d_4) = \frac{(0.125 * 0.250) + (0.125 * 0.250)}{\sqrt{(0.125^2 + 0.125^2)}.\sqrt{0.250^2 + 0.250^2 + 0.602^2}} = \frac{0.063}{0.177 * 0.698} = 0.510$$

Thus, Reqd. Ranking: 1: d_2; 2: d_4, 3: d_1, 4: d_3

P. T. O.

Q. 5> (i) Define a champion list for a term.

Ans. The idea of *champion lists* is to precompute, for each term *t* in the dictionary, the set of the *r* documents with the highest weights for *t*; the value of *r* is chosen in advance. For tf-idf weighting, these would be the *r* documents with the highest tf values for term *t*. We call this set of *r* documents the *champion list* for term *t*.

(ii) How is a champion list useful in IR?

Ans. Now, given a query *q* we create a set *A* as follows: we take the union of the champion lists for each of the terms comprising *q*. We can now restrict cosine computation to only the documents in *A,* and not the entire corpus.

(iii) What is an alternate term for a champion list? **(1 + 1 + 1 = 3)**

Ans. Fancy lists OR top docs.

Q. 6> Consider two queries for which there are 4 and 6 relevant documents in the collection respectively. Contrast two systems run on this collection. Their top 10 results are judged for relevance as follows (the leftmost item is the top ranked search result):

System 1, Query 1: R N R R R N N N N N

System 1, Query 2: R R R R N N N R N R

System 2, Query 1: N N N N R R R N N R

System 2, Query 2: N N N N R R R R R R

(i) What is the MAP of each system? Which system has a higher MAP?

AP for System 1, Query 1 = (1 + 2/3 + 3/4 + 4/5)/4 = 0.804

AP for System 1, Query 2 = (1 + 1 + 1 + 1 + 5/8 + 6/10)/6 = 0.871

MAP for System 1 = (0.804 + 0.871)/2 = **0.838**

AP for System 2, Query 1 = (1/5 + 2/6 + 3/7 + 4/10)/4 = 0.340

AP for System 2, Query 2 = (1/5 + 2/6 + 3/7 + 4/8 + 5/9 + 6/10)/6 = 0.436

MAP for System 2 = (0.340 + 0.436)/2 = **0.388**

System 1 has a higher map.

(ii) What does the result say about what is important in getting a good MAP score?

Ans. For getting a good MAP score, a system must retrieve relevant pages higher up in the ranked list of retrieved documents.

(iii) How is R-precision of a system defined? What is the R-precision of each system here? Does it rank the systems in the same order as MAP? **(6 + 1 + 3 = 10)**

Ans. The precision at the top *Rel* documents returned, where *Rel* is the number of relevant documents for the query, is known as the R-precision of the system.

R-precision for System 1 = (3/4 + 4/6)/2 = **0.708**

R-precision for System 2 = (0 + 2/6)/2 = **0.167**

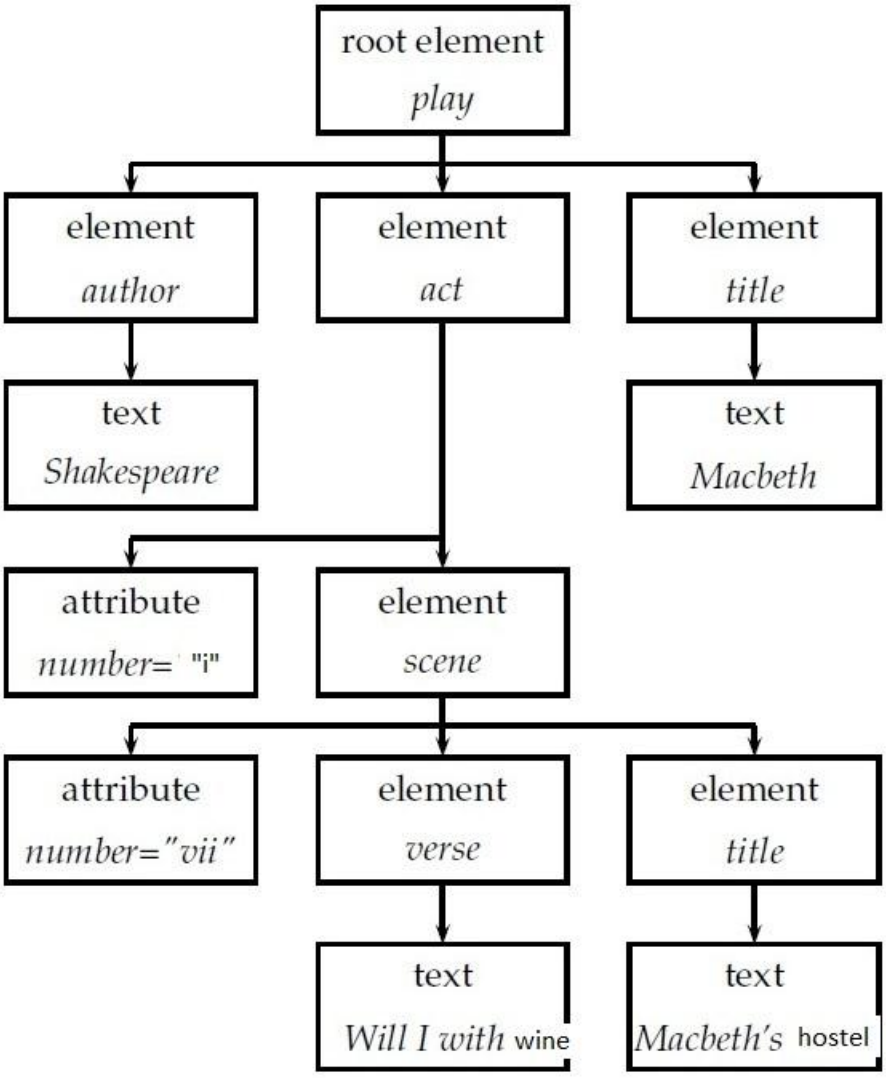
Yes, R-precision ranks the systems in the same order as MAP.

Q. 7> (i) Draw the XML DOM for the document below.

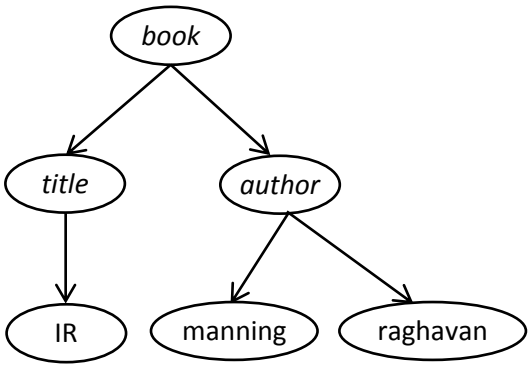
```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
<act number="i">
<scene number="vii">
<title>Macbeth's hostel</title>
<verse>Will I with wine</verse>
</scene>
</act>
</play>
```

Ans.

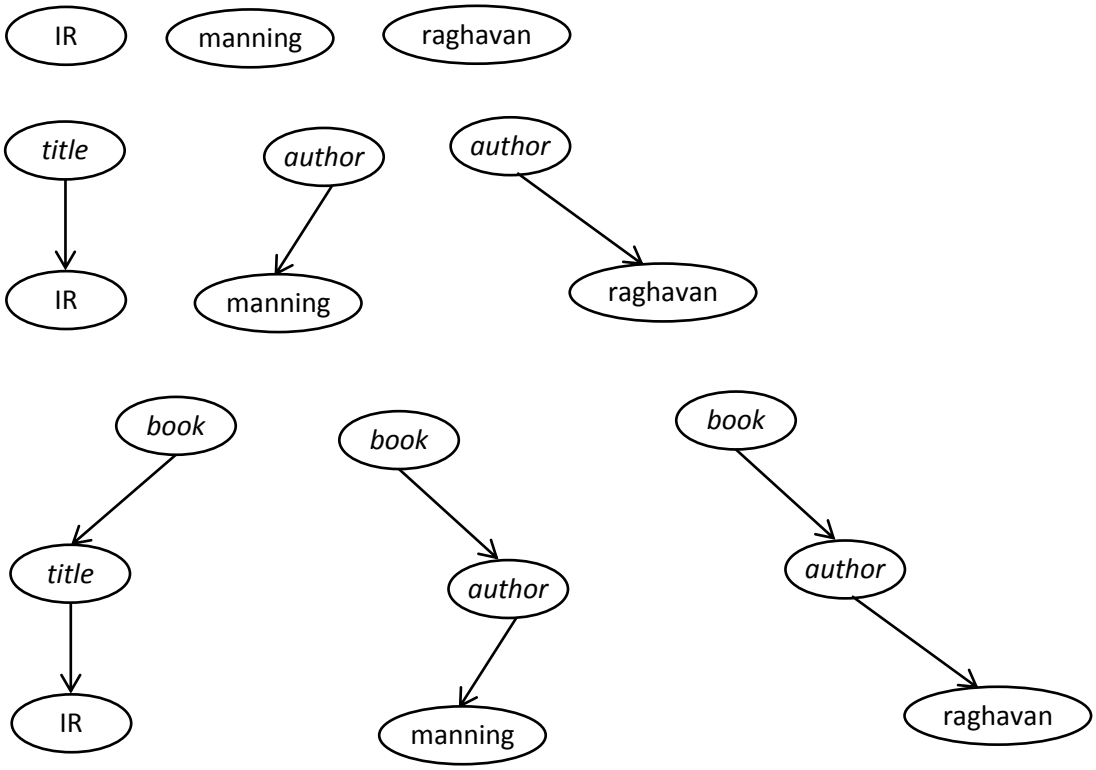
P. T. O.



(ii) What are the structural terms (represented as lexicalized subtrees) of the XML document below?



Ans. The **NINE structural terms** (represented as lexicalized subtrees) of the XML document are:



(iii) Name the following (expand abbreviations wherever applicable):

(a) Most popular format for XML queries

Ans. NEXI (Narrowed Extended XPath I)

(b) Evaluation forum for XML

Ans. INEX (INitiative for the Evaluation of XML retrieval)

(c) Two types of information needs (or topics) at (b)

Ans. Content-Only (CO) topics and Content-And-Structure (CAS) topics

(d) Two orthogonal dimensions of relevance assessment in XML **(3 + 3 + 4 = 10)**

Ans. Component coverage and Topical relevance