Information Retrieval (CS60092)
Mid-semester examination, Autumn 2013 – 2014

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

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

Q. 1>(a) Assume the word *cricket* has the following postings list:
cricket: 274, 287, 288, 300, 420
What is the variable-byte gap encoding (binary) for the above sequence?
How many bytes are required? How many bytes would have been required if the original postings list was stored (assume that one only needs to store the doc-ids)?

Soln. In variable byte gap encoding, the first doc-id is left unchanged.

274 in binary:100010010 -> split into groups of 7 -> 0000010 0010010
For first bit, 1 for last byte and 0 otherwise -> 00000010 10010010

Gaps = 13, 1, 12, 120
Binary = 1101, 1, 1100, 1111000
Padded and first bit set -> 10001101, 10000001, 10001100, 11111000

Thus, final encoding: 00000010 10010010 10001101 10000001 10001100 11111000 Ans.

6 bytes are required. Ans.

Doc-ids in binary: 100010010, 100011111, 100100000,100101100,110100100. All the numbers require 9 bits and hence 2 bytes each. [One cannot concatenate the bitstream to store in 45 bits, doc-id boundaries would be lost.]
Thus, **10 bytes** are required for storing the list without encoding. **Ans.**

(b) Decode the following variable byte gap encoding and report the actual postings list:
10000101 10000010 10000100 10000011 10000001

Nos. in decimal representation -> 5, 2, 4, 3, 1
No byte has starting bit 0.
Hence, final decoded list: 5, 7, 11, 14, 15. Ans.

(c) What are the largest gaps that can be encoded using one and two bytes?

One byte: 127 = $2^7 - 1$. **Two bytes: 16383** = $2^{14} - 1$ (14 out of the 16 bits can be used as payload). **Ans.**

(d) Can we derive Heaps' law from Zipf's law? Justify your answer. **[5 + 2 + 2 + 1 = 10]**

No, we cannot derive Heaps' Law from Zipf's Law. Zipf's Law states that the vocabulary is finite. Thus, it would follow that Heaps' Law does not hold [The definition of Zipf's Law states that the exponent is -1, which implies that the vocabulary is finite.]

Q. 2>Consider the four toy documents below as your corpus:
doc-1: *cricket is a great game*
doc-2: *a game is a good good sport*
doc-3: *all sport and cricket are great great great*
doc-4: *cricket is a sport and a game*
Assume the following stop list: *a, an, the, is, are, and, of, in, any, all, every, each*
We use a vector space model. Assume that the index of a term is determined by its order of encounter in the log (*cricket* gets position 1 in the vector). Using **simple TF-IDF** (use raw TF, multiplied by IDF(t) = $\log_{10}(N/\text{DF}(t))$) for **term weighting for both queries and documents**. Do **not** apply **any** normalization on the document weights. The issued query is *the great game of cricket.* Use the same TF-IDF vectors for both parts below. You are encouraged to neatly tabulate all values used, like term frequencies and term weights.

(a) Rank the documents using the simple overlap score. Show all steps of the computation.

(b) Rank the documents using the cosine similarity. Show all steps of the computation. **[5 + 5 = 10]**

Soln. Removing stop words, the terms to be indexed are: *cricket, great, game, good, sport*[in order of encounter]. DF = document frequency, tf = term frequency, tw = term weight

Word	DF(t)	IDF(t)	doc-1 tf	doc-1 tw	doc-2 tf	doc-2 tw	doc-3 tf	doc-3 tw	doc-4 tf	doc-4 tw	query-tf	query-tw
cricket	3	0.1249	1	0.1249	0	0	1	0.1249	1	0.1249	1	0.1249
great	2	0.3010	1	0.3010	0	0	3	0.9030	0	0	1	0.3010
game	3	0.1249	1	0.1249	1	0.1249	0	0	1	0.1249	1	0.1249
good	1	0.6021	0	0	2	1.2042	0	0	0	0	0	0
sport	3	0.1249	0	0	1	0.1249	1	0.1249	1	0.1249	0	0

Similarly, the query is also reduced to *great game cricket*

(a) Rank the documents using the simple overlap score. Show all steps of the computation.

The overlap score for a query-document pair is given by $$\text{Score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d}.$$

Thus, overlap-score(q, doc-1) = 0.1249 + 0.3010 + 0.1249 = 0.5508
overlap-score(q, doc-2) = 0 + 0 + 0.1249 = 0.1249
overlap-score(q, doc-3) = 0.1249 + 0.9030 + 0 = 1.0279
overlap-score(q, doc-4) = 0.1249 + 0 + 0.1249 = 0.2498

Thus, desired ranking: doc-3, doc-1, doc-4, doc-2. Ans.

(b) Rank the documents using the cosine similarity. Show all steps of the computation. **[5 + 5 = 10]**

For cosine similarity, the scoring function is given by $$\text{score}(q,d) = \frac{\vec{V}(q)\cdot\vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}$$

The query and document vectors are given by the respective term weight vectors in the table, reproduced again for convenience below.

Word	DF(t)	IDF(t)	doc-1 tf	doc-1 tw	doc-2 tf	doc-2 tw	doc-3 tf	doc-3 tw	doc-4 tf	doc-4 tw	query-tf	query-tw
cricket	3	0.1249	1	0.1249	0	0	1	0.1249	1	0.1249	1	0.1249
great	2	0.3010	1	0.3010	0	0	3	0.9030	0	0	1	0.3010
game	3	0.1249	1	0.1249	1	0.1249	0	0	1	0.1249	1	0.1249
good	1	0.6021	0	0	2	1.2042	0	0	0	0	0	0
sport	3	0.1249	0	0	1	0.1249	1	0.1249	1	0.1249	0	0

Then (**bold** represents vector),
$|\mathbf{V}(\text{doc-1})| = (0.1249^2 + 0.3010^2 + 0.1249^2 + 0^2 + 0^2)^{1/2} = (0.0156 + 0.0906 + 0.0156)^{1/2} = 0.3490$
$|\mathbf{V}(\text{doc-2})| = (0^2 + 0^2 + 0.1249^2 + 1.2042^2 + 0.1249^2)^{1/2} = (0.0156 + 1.4501 + 0.0156)^{1/2} = 1.2171$
$|\mathbf{V}(\text{doc-3})| = (0.1249^2 + 0.9030^2 + 0^2 + 0^2 + 0.1249^2)^{1/2} = (0.0156 + 0.8154 + 0.0156)^{1/2} = 0.9201$
$|\mathbf{V}(\text{doc-4})| = (0.1249^2 + 0^2 + 0.1249^2 + 0^2 + 0.1249^2)^{1/2} = (0.0156 + 0.0156 + 0.0156)^{1/2} = 0.2163$
$|\mathbf{V}(q)| = (0.1249^2 + 0.3010^2 + 0.1249^2 + 0^2 + 0^2)^{1/2} = (0.0156 + 0.0906 + 0.0156)^{1/2} = 0.3490$

cos-sim-score(q, doc-1) = (0.1249*0.1249 + 0.3010*0.3010 + 0.1249*0.1249)/(0.3490*0.3490) = 1.0000
cos-sim-score(q, doc-2) = (0.1249*0.1249)/(0.3490*1.2171) = 0.0367
cos-sim-score(q, doc-3) = (0.1249*0.1249 + 0.3010*0.9030)/(0.3490*0.9201) = 0.8950
cos-sim-score(q, doc-4) = (0.1249*0.1249 + 0.1249*0.1249)/(0.3490*0.2163) = 0.4133

Thus, desired ranking: doc-1, doc-3, doc-4, doc-2. Ans.

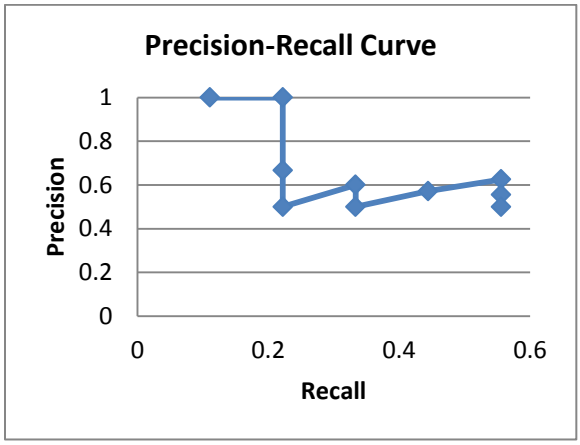
Q. 3>A retrieval system produced the following ranked list (only relevance judgments are shown) in response to a query: 1, 1, 0, 0, 1, 0, 1, 1, 0, 0. Assume the number of relevant documents in the collection to be nine.

(a) Compute the F-score for the query.

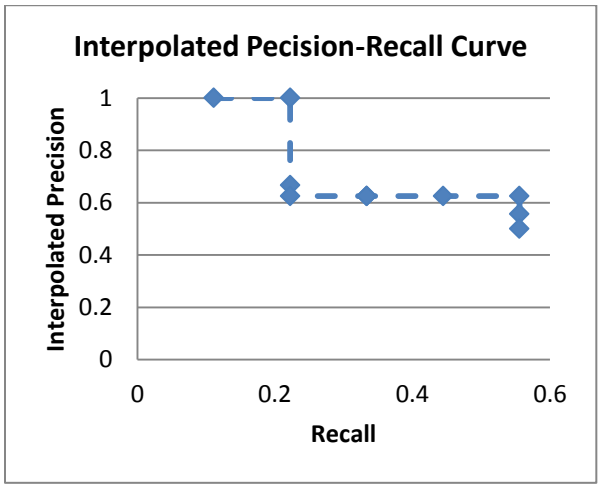
Soln. Precision = 5/10 = 0.5
Recall = 5/9 = 0.5556
F-score = (2*0.5*0.5556)/(0.5 + 0.5556) = **0.5263 Ans.**

(b) Plot the precision-recall curve for this query.

Rank	Relevance	Recall	Precision	Interp. Precision
1	1	0.1111	1.0000	1.0000
2	1	0.2222	1.0000	1.0000
3	0	0.2222	0.6667	0.6667
4	0	0.2222	0.5000	0.6667
5	1	0.3333	0.6000	0.6250
6	0	0.3333	0.5000	0.6250
7	1	0.4444	0.5714	0.6250
8	1	0.5556	0.6250	0.6250
9	0	0.5556	0.5556	0.5556
10	0	0.5556	0.5000	0.5000



(c) Plot the interpolated precision-recall curve for this query.



(d) Now assume the same system produces the list: 0, 0, 1, 1, 1, 0, 1, 0, 1, 0 for a second query. Assume the number of relevant documents for this query to be seven. Compute mean R-precision.

R-precision for query 1 (|Rel| = 9) = 5/9 = 0.5556
R-precision for query 2 (|Rel| = 7) = 4/7 = 0.5714
Mean R-precision = **0.5635 Ans.**

(e) Compute MAP for this binary relevance system. **[1 + 3 + 3 + 1.5 + 1.5 = 10]**

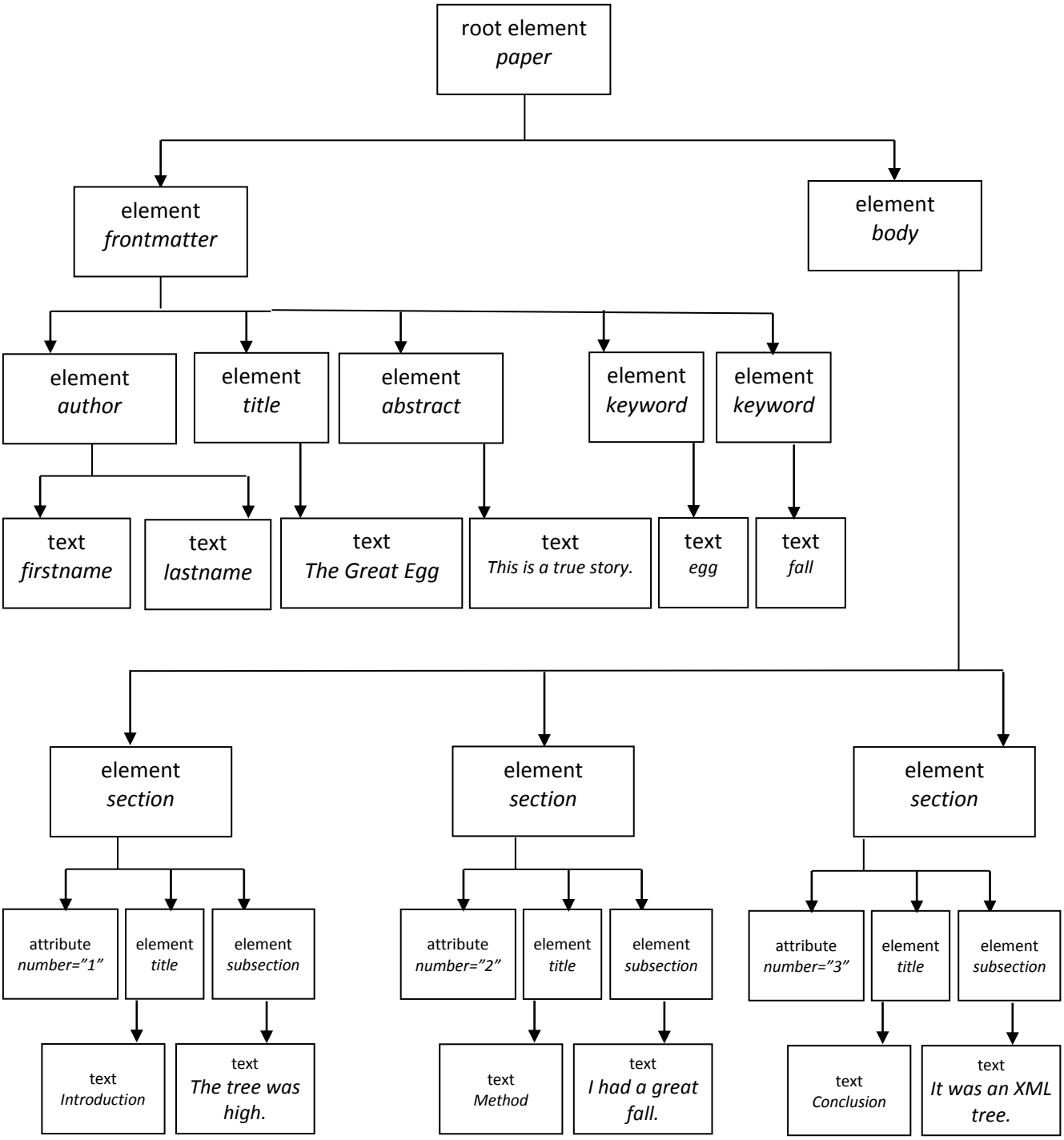
AP for query 1 = 1/9(1/1 + 2/2 + 3/5 + 4/7 + 5/8) = 0.4218
AP for query 2 = 1/7(1/3 + 2/4 + 3/5 + 4/7 + 5/9) = 0.3658
Thus, MAP = (0.4218 + 0.3658)/2 = **0.3938. Ans.**

P. T. O.

Q. 4>(a) Provide the tree representation of the following XML document:

```
<paper>
<frontmatter>
<author>
<firstname>Humpty</firstname>
<lastname>Dumpty</lastname>
</author>
<title>The Great Egg</title>
<abstract>This is a true story.</abstract>
<keyword>egg</keyword>
<keyword>fall</keyword>
</frontmatter>
<body>
<section number="1">
<title>Introduction</title>
<subsection number="1.1">I sat on a tree.</subsection>
</section>
<section number="2">
<title>Method</title>
<subsection number="2.1">The tree was high.</subsection>
<subsection number="2.2">I had a great fall!</subsection>
</section>
<section number="3">
<title>Conclusion</title>
<subsection number="3.1">It was an XML tree.</subsection>
</section>
</body>
</paper>
```

P. T. O.



(b) Expand NEXI and INEX. **[8 + 2 = 10]**

NEXI: Narrowed Extended XPath I
INEX: Initiative for XML Retrieval

Q. 5>(a) The postings lists for *cricket, football, swimming* and *tennis* contain 35, 77, 12 and 18 documents respectively. What is the best order for processing the query that is a conjunction of all the four terms? Use parentheses to show the order clearly.

Soln. (*football* AND (*cricket* AND (*swimming* AND *tennis*)))

(b) If the document frequencies are assumed to be *w, x, y* and *z*, and the corpus has *N* documents in all, what is the time complexity for processing the query *(cricket OR football) AND NOT (swimming OR tennis))*?

Soln.*O(N)*. We can always intersect in *O(qN)* where *q* is the no. of query terms and *N* the number of documents, so the intersection time is linear in the no. of documents and query terms. Since the tightest bound for the size of the results list is *N*, the number of all documents, we cannot do better than *O(N)*.

(c) Which of stemming and lemmatization can be expected to produce a larger lexicon? Why?
Soln. Stemming produces a larger lexicon. In principle, stemming leaves certain verb forms of irregular verbs like *brought, wrote, struck* as they are, and these get added to the lexicon. However, a lemmatizer is expected to map all of these to their respective base forms *bring*, *write* and *strike*. The effect of stemming and lemmatization is the same (from the point of vocabulary size) for regular verb forms like *burning, duplicated, loves*even though the lexicon entries may differ.

(d) Name three popular stemming algorithms.

Soln. Porter Stemmer, Lovins stemmer and Paice stemmer.

(e) Can stemming decrease precision? Justify your answer.

Soln. Yes, stemming can decrease precision. Stemming can increase the number of retrieved documents without increasing the number of relevant documents.

(f) Assume the following postings lists for *cake* and *pie*:
cake: 3, 5, 7, 10, 13, 18, 39, 53 [Skips inserted from 3 to 10 and 10 to 39]
pie: 2, 12, 15, 39, 43, 45, 49 [Skips from 2 to 39 and 39 to 49]
List the pairwise doc-id comparisons that are avoided due to the presence of skip pointers.

Soln. All comparisons (without skips):
(3, 2), (3, 12), (5, 12), (7, 12), (10, 12), (13, 12), (13, 15), (18, 15), (18, 39), (39, 39), (53, 43), (53, 45), (53, 49), end.

Comparisons (with skips):
(3, 2), (3, 39), (3, 12), (3, 12), (10, 12), (5, 12), (10, 12), (39, 12), (13, 12), (13, 12), (13, 15), (18, 15), (18, 39), (39, 39), (53, 49), (53, 43), (53, 49), end.

Thus, avoided comparisons: **(3, 12), (5, 12), (7, 12), (53, 45)**. **Ans.**
Note that the (53, 43) comparison is required anyway for deciding whether to skip.

(g) Enumerate the permuterm vocabulary for *cake*.

Soln.*cake$, ake$c, keca, ecak, $cake*

(h) Without computations, state the Levenshtein distance between *calm* and *slam*.

Ans. 3 (one substitution, deletion and insertion each) *calm -> salm -> sam -> slam*. (There are other possible routes, but the distance remains 3).

(i) What is the Jaccard coefficient between *claim* and *clam* using character bigrams?

Soln. Set 1 = *cl, la, ai, im, m$*
Set 2 = *cl, la, am, m$*
|Set 1 ∩ Set 2| = 3
|Set 1 U Set 2| = 6
Jaccard = 3/6 = **0.5 Ans.**

(j) Name one real application of *k*-gram character indexes. **[1 x 10 = 10]**

Ans. Context-sensitive query spelling correction. (simply spelling correction is also fine).
