Homework 1  (10’)

Page 9: Exercise 1.2; Exercise 1.3
Page 12: Exercise 1.6
Page 13: Exercise 1.8; Exercise 1.10
Page 33: Exercise 2.1; Exercise 2.3
Page 36: Exercise 2.7
Page 41: Exercise 2.9
Page 51: Exercise 3.2; Exercise 3.3;
Page 57: Exercise 3.10
Page 73: Exercise 4.4
Page 75: Exercise 4.11
Page 87: Exercise 5.3
Page 95: Exercise 5.5

Exercise 1.2  (0.5’)
Consider these documents:
Doc 1 breakthrough drug for schizophrenia
Doc 2 new schizophrenia drug
Doc 3 new approach for treatment of schizophrenia
Doc 4 new hopes for schizophrenia patients
a. Draw the term-document incidence matrix for this document collection.
b. Draw the inverted index representation for this collection, as in Figure 1.3 (page 6).

a. Term-document incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Doc1</th>
<th>Doc2</th>
<th>Doc3</th>
<th>Doc4</th>
</tr>
</thead>
<tbody>
<tr>
<td>approach</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>breakthrough</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>drug</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>for</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>hopes</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>new</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>of</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>schizophrenia</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>treatment</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

b. Inverted index representation for this collection (change the order between “hopes” and “for”)

<table>
<thead>
<tr>
<th>Term</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>approach</td>
<td>3</td>
</tr>
<tr>
<td>breakthrough</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise 1.3  (0.5’)
For the document collection shown in Exercise 1.2, what are the returned results for these queries:
a. schizophrenia AND drug
b. for AND NOT(drug OR approach)

a. Doc1, Doc2
b. Doc4

Exercise 1.6  (1’)
We can use distributive laws for AND and OR to rewrite queries.

a. Show how to rewrite the query in Exercise 1.5 into disjunctive normal form using the distributive laws.
b. Would the resulting query be more or less efficiently evaluated than the original form of this query?
c. Is this result true in general or does it depend on the words and the contents of the document collection?

a. 
(Brutus OR Caesar) AND NOT (Antony OR Cleopatra)
= (Brutus OR Caesar) AND NOT Antony AND NOT Cleopatra
= (Brutus AND (NOT Antony) AND(NOT Cleopatra)) OR (Caesar AND (NOT Antony) AND(NOT Cleopatra))

b. The resulting query would be more efficiently evaluated than the original form of this query.
c. It depends on the words and the contents of the document collection.

**Exercise 1.8 (0.5’)**

If the query is:
e. friends AND romans AND (NOT countrymen)

how could we use the frequency of countrymen in evaluating the best query evaluation order? In particular, propose a way of handling negation in determining the order of query processing.

We always use the frequency of countrymen to evaluate the best query evaluation order.

**Exercise 1.10 (0.5’)**

Write out a postings merge algorithm, in the style of Figure 1.6 (page 11), for an x OR y query.

```
UNION(p1, p2)
   answer<--<<
   while p1 != NIL or p2 != NIL do
      if p1 = NIL
         ADD(answer, docID(p2))
p2 <- next(p2)
      else if p2 = NIL
         ADD(answer, docID(p1))
p1 <- next(p1)
      else
         if docID(p1) = docID(p2)
            ADD(answer, docID(p1))
p1 <- next(p1)
p2 <- next(p2)
         elseif docID(p1) < docID(p2)
            ADD(answer, docID(p1))
p1 <- next(p1)
         else
            ADD(answer, docID(p2))
p2 <- next(p2)
      return answer
```

**Exercise 2.1 (0.5’)**

Are the following statements true or false?
In a Boolean retrieval system, stemming never lowers precision.
In a Boolean retrieval system, stemming never lowers recall.
Stemming increases the size of the vocabulary.
Stemming should be invoked at indexing time but not while processing a query.

Exercise 2.3 (0.5’)
The following pairs of words are stemmed to the same form by the Porter stemmer. Which pairs would you argue shouldn’t be conflated. Give your reasoning.

a. abandon/abandonment
b. absorbency/absorbent
c. marketing/markets
d. university/universe
e. volume/volumes

c. marketing/market should not be conflated
d. university/universeshouldnot be conflated

Exercise 2.7 (1’)
Consider a postings intersection between this postings list, with skip pointers:
3 5 9 15 24 39 60 68 75 81 84 89 92 96 97 100 115
and the following intermediate result postings list (which hence has no skip pointers):
3 5 89 95 97 99 100 101
Trace through the postings intersection algorithm in Figure 2.10 (page 37).

a. How often is a skip pointer followed (i.e., p1 is advanced to skip(p1))?
b. How many postings comparisons will be made by this algorithm while intersecting the two lists?
c. How many postings comparisons would be made if the postings lists are intersected without the use of skip pointers?

a. 1 time, 24→75
b. 18
   3=3 5=5 9<89 15<89 24<89 75<89 92<89 81<89 84<89 89<89 95<92 95<115
   95<96 97>96 97=97 99<100 100=100 101<115
c. 19
   3=3 5=5 89<9 89<15 89<24 89<39 89<60 89<68 89<75 89<81 89<84 89<89 95<92
Exercise 2.9  (0.5’)
Shown below is a portion of a positional index in the format: term: doc1: hposition1, position2, . . . ; doc2: hposition1, position2, . . . ; etc.

angels: 2: h36,174,252,651i; 4: h12,22,102,432i; 7: h17i;
fools: 2: h1,17,74,222i; 4: h8,78,108,458i; 7: h3,13,23,193i;
fear: 2: h87,704,722,901i; 4: h13,43,113,433i; 7: h18,328,528i;
in: 2: h3,37,76,444,851i; 4: h10,20,110,470,500i; 7: h5,15,25,195i;
rush: 2: h2,66,194,321,702i; 4: h9,69,149,429,569i; 7: h4,14,404i;
to: 2: h47,86,234,999i; 4: h14,24,774,944i; 7: h199,319,599,709i;
tread: 2: h57,94,333i; 4: h15,35,155i; 7: h20,320i;
where: 2: h67,124,393,1001i; 4: h11,41,101,421,431i; 7: h16,36,736i;

Which document(s) if anymatch each of the following queries, where each expression within quotes is a phrase query?

a.  “fools rush in”

b.  “fools rush in” AND “angels fear to tread”

Exercise 3.2  (0.5’)
Write down the entries in the permuterm index dictionary that are generated by the term mama.

mama$
ama$m
ma$m
a$mam
$mama

Exercise 3.3  (0.5’)
If you wanted to search for s*ng in a permuterm wildcard index, what key(s) would one do the lookup on?

ng$s*

Exercise 3.10  (0.5’)
Compute the Jaccard coefficients between the query bord and each of the terms in Figure 3.7 that contain the bigram or.
<table>
<thead>
<tr>
<th>bord</th>
<th>border</th>
<th>lord</th>
<th>morbid</th>
<th>sordid</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/5</td>
<td>1/2</td>
<td>1/7</td>
<td>1/3</td>
<td></td>
</tr>
</tbody>
</table>

**Exercise 4.4 (0.5’)**
For \( n = 2 \) and \( 1 \leq T \leq 30 \), perform a step-by-step simulation of the algorithm in Figure 4.7. Create a table that shows, for each point in time at which \( T = 2^k \) tokens have been processed (\( 1 \leq k \leq 15 \)), which of the three indexes \( I_0, \ldots, I_3 \) are in use. The first three lines of the table are given below.

\[
\begin{array}{cccc}
I_3 & I_2 & I_1 & I_0 \\
2 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 \\
6 & 0 & 0 & 1 \\
8 & 0 & 0 & 1 \\
10 & 0 & 1 & 0 \\
12 & 0 & 1 & 0 \\
14 & 0 & 1 & 1 \\
16 & 0 & 1 & 1 \\
18 & 1 & 0 & 0 \\
20 & 1 & 0 & 1 \\
22 & 1 & 0 & 1 \\
24 & 1 & 0 & 1 \\
26 & 1 & 1 & 0 \\
28 & 1 & 1 & 0 \\
30 & 1 & 1 & 1 \\
\end{array}
\]

**Exercise 4.11 (1’)**
Apply MapReduce to the problem of counting how often each term occurs in a set of files. Specify map and reduce operations for this task. Write down an example along the lines of Figure 4.6. (should follow the example in Figure 4.6).

**Method 1:**

**Schema:**
map: input \( \rightarrow \) list(k, v)
reduce: list(k, v) \( \rightarrow \) output

**Instantiation of the schema for term counting**
map: a set of files \( \rightarrow \) list(term, 1)
reduce: \( <(\text{term1}, 1), (\text{term2}, 1), (\text{term3}, 1), \ldots> \rightarrow \) list(term, total count)

**Example for term counting**
map: d1:I hear, I forget. d2:I see, I remember. \( \rightarrow <I, 1><\text{hear}, 1><I, 1><\text{forget}, 1> \)
Method 2:
Schema:
map: input → list(k, v)
reduce: list(k, v) → output

Instantiation of the schema for term counting
map: a set of files → list(term, count in one file)
reduce: <(term1, count1), (term2, count2), (term3, count3)...> → list(term, total count)

Example for term counting
map: d1: I hear, I forget.
d2: I see, I remember.
reduce: <I, 1><see, 1><remember 1>

Exercise 5.3 (0.5’)
Estimate the time needed for term lookup in the compressed dictionary of Reuters-RCV1 with block sizes of k = 4, k = 8, and k = 16. What is the slowdown compared with k = 1?

Average steps needed to look up term is
log(N/k) - 1 + k/2

For Reuters-RCV1, N=400000

<table>
<thead>
<tr>
<th>K</th>
<th>Average steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>17.6</td>
</tr>
<tr>
<td>8</td>
<td>18.6</td>
</tr>
<tr>
<td>16</td>
<td>21.6</td>
</tr>
</tbody>
</table>

Exercise 5.5 (1’)
Compute variable byte and γ codes for the postings list <777, 17743, 294068, 31251336>.
Use gaps instead of doclDs where possible. Write binary codes in 8-bit blocks.
(double-check with others’ answer)
<table>
<thead>
<tr>
<th>vb</th>
<th>777</th>
<th>17743</th>
<th>294068</th>
<th>31251336</th>
</tr>
</thead>
<tbody>
<tr>
<td>gaps</td>
<td>777</td>
<td>16966</td>
<td>276325</td>
<td>30957268</td>
</tr>
<tr>
<td>vb</td>
<td>00000110, 10001001</td>
<td>00000001, 00000100, 11000110</td>
<td>00010000, 01101110, 11100101</td>
<td>00001110, 01100001, 00111101, 11010100</td>
</tr>
<tr>
<td>γ</td>
<td>111111111, 100001001</td>
<td>111111111111, 1110, 00001001000110</td>
<td>111111111111111, 1110, 0000110111010110101</td>
<td>11111111111111111111, 11110, 11011000010111011010100</td>
</tr>
</tbody>
</table>