Bilkent University - CS533
Homework 2

Shatlyk Ashyralyyev

1 Question 1

Consider the following search results for two queries Q1 and Q2 (the documents are ranked in the given order, the relevant documents are shown in bold).

Q1: **D1**, **D2**, D3, **D4**, D5, **D6**, D7, D8, D9, D10.

Q2: **D1**, D2, **D3**, D4, D5, **D6**, D7, D8, **D9**, D10.

For Q1 and Q2 the total number of relevant documents are, respectively, 5 and 8 (for example for Q1 one of the relevant documents is not retrieved).

a. Using the TREC interpolation rule, in a table give the precision value for the 11 standard recall levels 0.0, 0.1, 0.2, ... 1.0. Please also draw the corresponding recall-precision graph as shown in the first figure of TREC-6 Appendix A (its link is available on the course web site).

Hint. "Interpolated" means that, for example, precision at recall 0.10 (i.e., after 10
Please do this for each query separately and obtain one table for both queries using the average of two values at each recall point.

b. Find R-Precision (TREC-6 Appendix A for definition) for Query1 and Query2.

c. Find MAP for these queries.

d. Calculate precision and recall values @10 using the concepts of TP, FP, TN, FN: true positive, false positive, true negative, and false negative.

(a)

Given a query *Q1* with 10 retrieved documents for the search of the query. While, the total number of relevant documents is 5, relevant documents that were retrieved are shown in bold.

Q1 : **D1**, **D2**, D3, **D4**, D5, **D6**, D7, D8, D9, D10.
Table 1 shows the recall values and precision values for top k results of *Q1*.

Table 1: Recall and Precision values for top k results of Q1.

k	k^{th} result	relevant/irrelevant	precision	recall
1	**D1**	relevant	1/1=1.000	1/5=0.200
2	**D2**	relevant	2/2=1.000	2/5=0.400
3	D3	irrelevant	2/3=0.667	2/5=0.400
4	**D4**	relevant	3/4=0.750	3/5=0.600
5	D5	irrelevant	3/5=0.600	3/5=0.600
6	**D6**	relevant	4/6=0.667	4/5=0.800
7	D7	irrelevant	4/7=0.571	4/5=0.800
8	D8	irrelevant	4/8=0.500	4/5=0.800
9	D9	irrelevant	4/9=0.444	4/5=0.800
10	D10	irrelevant	4/10=0.400	4/5=0.800

Given a query *Q2* with 10 retrieved documents for the search of the query. While, the total number of relevant documents is 8, relevant documents that were retrieved are shown in bold.

Q2 : **D1**, D2, **D3**, D4, D5, **D6**, D7, D8, **D9**, D10.
Table 2 shows the recall values and precision values for top k results of *Q2*.

Table 2: Recall and Precision values for top k results of Q2.

k	k^{th} result	relevant/irrelevant	precision	recall
1	**D1**	relevant	1/1=1.000	1/8=0.125
2	D2	irrelevant	1/2=0.500	1/8=0.125
3	**D3**	relevant	2/3=0.667	2/8=0.250
4	D4	irrelevant	2/4=0.500	2/8=0.250
5	D5	irrelevant	2/5=0.400	2/8=0.250
6	**D6**	relevant	3/6=0.500	3/8=0.375
7	D7	irrelevant	3/7=0.429	3/8=0.375
8	D8	irrelevant	3/8=0.375	3/8=0.375
9	**D9**	relevant	4/9=0.444	4/8=0.500
10	D10	irrelevant	4/10=0.400	4/8=0.500

Table 3 shows the precision values of *Q1*, *Q2* and their average for 11 standard recall levels. Figure 1 shows the recall-precision graph for *Q1*, *Q2* and their average for 11 standard recall levels.

Table 3: Precision values for 11 standard recall levels.

Recall level	Q1 Precision	Q2 Precision	Average Precision
0.0	1.000	1.000	1.000
0.1	1.000	1.000	1.000
0.2	1.000	0.667	0.834
0.3	1.000	0.500	0.750
0.4	1.000	0.444	0.722
0.5	0.750	0.444	0.597
0.6	0.750	0.000	0.375
0.7	0.667	0.000	0.334
0.8	0.667	0.000	0.334
0.9	0.000	0.000	0.000
1.0	0.000	0.000	0.000

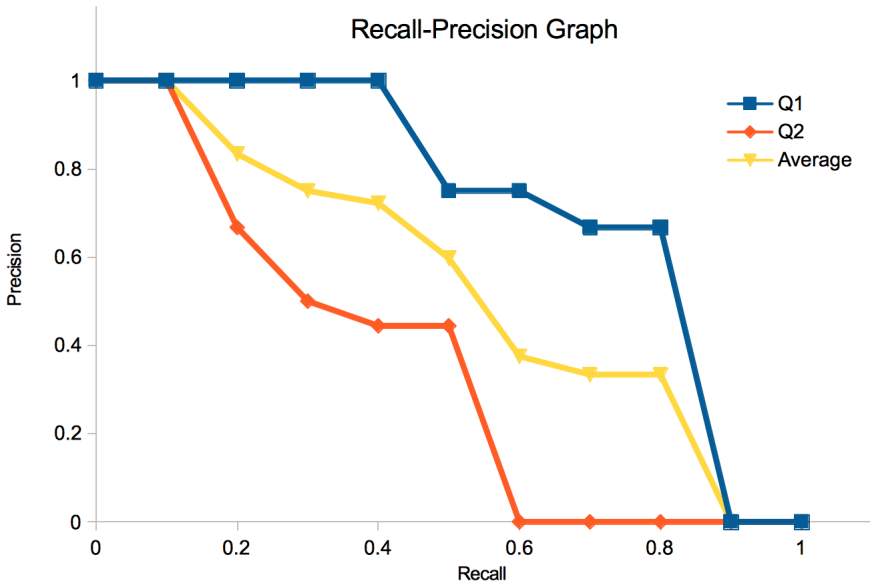

Figure 1: Recall-Precision Graph for 11 standard recall values.

(b)

From the definition, *R-Precision* is the precision after R documents have been retrieved, where R is the number of relevant documents for given query.

R-Precision for Query1 = Precision after 5 documents have been retrieved = $\frac{3}{5} = 0.6$

R-Precision for Query2 = Precision after 8 documents have been retrieved = $\frac{3}{8} = 0.375$

(c)

Average Precision for Query1:

$$\frac{\sum \text{precision after each relevant document retrieved}}{\text{total number of relevant documents}} = \frac{1+1+0.75+0.667}{5} = 0.6834$$

Average Precision for Query2:

$$\frac{\sum \text{precision after each relevant document retrieved}}{\text{total number of relevant documents}} = \frac{1+0.667+0.5+0.444}{8} = 0.3264$$

Mean Average Precision (MAP) for these queries:

$$\frac{\text{Average Precision for Query1}+\text{Average Precision for Query2}}{2} = \frac{0.6834+0.3264}{2} = 0.5049$$

(d)

Recall the definition of Precision and Recall using TP, FP, TN, FN. They are given in Table 4.

Table 4: Definition of precision and recall using TP, FP, TN, FN.

	Relevant	Irrelevant
Retrieved	true positives (TP)	false positives (FP)
Not Retrieved	false negative (FN)	true negatives (TN)

Using Table 4, we define Precision = $\frac{TP}{(TP+FP)}$ and Recall = $\frac{TP}{(TP+FN)}$.
For Query1, $TP = 4$, $FP = 6$, $FN = 1$. Therefore, $P = \frac{4}{4+6} = \frac{4}{10} = 0.4$ and $R = \frac{4}{4+1} = \frac{4}{5} = 0.8$.
For Query2, $TP = 4$, $FP = 6$, $FN = 4$. Therefore, $P = \frac{4}{4+6} = \frac{4}{10} = 0.4$ and $R = \frac{4}{4+4} = \frac{4}{8} = 0.5$.

All of the values above agree with the values calculated using traditional definition of Precision and Recall.

2 Question 2

Consider the following document by term binary D matrix for m= 6 documents (rows), n= 6 terms (columns).

$$D = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Consider the problem of constructing a document by document similarity, S, matrix. How many similarity coefficients will be calculated using the following methods? For each case explain your answer briefly: give exact numbers for each document and briefly explain how you came up with those numbers.

a. Straightforward approach (using document vectors) -the 1st method discussed in the class-.

b. Using term inverted indexes.

c. Calculate the similarity values of all documents using the cosine and Dice coefficients. Calculate the match (correlation) between these two measures by using the Kendall's tau measure. Please briefly show your calculations.

(a)

Straightforward approach calculates the similarities between all possible document pairs. Let's define $S(d_i, d_j)$ to be the similarity between document d_i and document d_j. We assume that the similarity is symmetric, i.e., $S(d_i, d_j) = S(d_j, d_i)$. Therefore, we may calculate the similarities of only half of the possible pairs. Moreover, we skip the calculations of the similarities of the documents with themselves. That means, $\forall$i, $S(d_i, d_i) = \textit{maximum possible similarity score} = 1.0$. After all these observations, let's construct the similarity matrix S:

$$S = \begin{bmatrix} S(d_1,d_1)=1 & \boldsymbol{S(d_1,d_2)} & \boldsymbol{S(d_1,d_3)} & \boldsymbol{S(d_1,d_4)} & \boldsymbol{S(d_1,d_5)} & \boldsymbol{S(d_1,d_6)} \\ - & S(d_2,d_2)=1 & \boldsymbol{S(d_2,d_3)} & \boldsymbol{S(d_2,d_4)} & \boldsymbol{S(d_2,d_5)} & \boldsymbol{S(d_2,d_6)} \\ - & - & S(d_3,d_3)=1 & \boldsymbol{S(d_3,d_4)} & \boldsymbol{S(d_3,d_5)} & \boldsymbol{S(d_3,d_6)} \\ - & - & - & S(d_4,d_4)=1 & \boldsymbol{S(d_4,d_5)} & \boldsymbol{S(d_4,d_6)} \\ - & - & - & - & S(d_5,d_5)=1 & \boldsymbol{S(d_5,d_6)} \\ - & - & - & - & - & S(d_6,d_6)=1 \end{bmatrix}$$

When the straightforward approach is used, all bold elements of the matrix S should be calculated. The general formula is $\frac{m*(m-1)}{2}$, where m is the number of documents. Therefore, using the straightforward approach we should calculcate $\frac{6*(6-1)}{2} = 15$ document-by-document similarity coefficients.

(b)

First of all, let's construct the term inverted indexes.

$$\begin{array}{lll}
t_1 & \Longrightarrow & < d_1, 1 >, \ < d_3, 1 > \\
t_2 & \Longrightarrow & < d_2, 1 >, \ < d_4, 1 > \\
t_3 & \Longrightarrow & < d_1, 1 >, \ < d_3, 1 >, \ < d_5, 1 > \\
t_4 & \Longrightarrow & < d_2, 1 >, \ < d_3, 1 >, \ < d_4, 1 > \\
t_5 & \Longrightarrow & < d_1, 1 >, \ < d_2, 1 >, \ < d_5, 1 >, \ < d_6, 1 > \\
t_6 & \Longrightarrow & < d_5, 1 >, \ < d_6, 1 >
\end{array}$$

- Consider d_1. Terms that appear in d_1: $\{t_1, t_3, t_5\}$. Their inverted indexes:

$$\begin{array}{lll}
t_1 & \Longrightarrow & \boldsymbol{< d_1, 1 >}, \ < d_3, 1 > \\
t_3 & \Longrightarrow & \boldsymbol{< d_1, 1 >}, \ < d_3, 1 >, \ < d_5, 1 > \\
t_5 & \Longrightarrow & \boldsymbol{< d_1, 1 >}, \ < d_2, 1 >, \ < d_5, 1 >, \ < d_6, 1 >
\end{array}$$

  Using the inverted indexes we can see that d_1 has common terms with d_2, d_3, d_5 and d_6. Therefore, $S(d_1, d_2)$, $S(d_1, d_3)$, $S(d_1, d_5)$ and $S(d_1, d_6)$ similarity coefficients should be calculated.

- Consider d_2. Terms that appear in d_2: $\{t_2, t_4, t_5\}$. Their inverted indexes:

$$\begin{array}{lll}
t_2 & \Longrightarrow & \boldsymbol{< d_2, 1 >}, \ < d_4, 1 > \\
t_4 & \Longrightarrow & \boldsymbol{< d_2, 1 >}, \ < d_3, 1 >, \ < d_4, 1 > \\
t_5 & \Longrightarrow & < d_1, 1 >, \ \boldsymbol{< d_2, 1 >}, \ < d_5, 1 >, \ < d_6, 1 >
\end{array}$$

  Using the inverted indexes we can see that d_2 has common terms with d_1, d_3, d_4, d_5 and d_6. Therefore, $S(d_2, d_1)$, $S(d_2, d_3)$, $S(d_2, d_4)$, $S(d_2, d_5)$ and $S(d_2, d_6)$ similarity coefficients should be calculated.

- Consider d_3. Terms that appear in d_3: $\{t_1, t_3, t_4\}$. Their inverted indexes:

$$\begin{array}{lll}
t_1 & \Longrightarrow & < d_1, 1 >, \ \boldsymbol{< d_3, 1 >} \\
t_3 & \Longrightarrow & < d_1, 1 >, \ \boldsymbol{< d_3, 1 >}, \ < d_5, 1 > \\
t_4 & \Longrightarrow & < d_2, 1 >, \ \boldsymbol{< d_3, 1 >}, \ < d_4, 1 >
\end{array}$$

  Using the inverted indexes we can see that d_3 has common terms with d_1, d_2, d_4 and d_5. Therefore, $S(d_3, d_1)$, $S(d_3, d_2)$, $S(d_3, d_4)$ and $S(d_3, d_5)$ similarity coefficients should be calculated.

- Consider d_4. Terms that appear in d_4: $\{t_2, t_4\}$. Their inverted indexes:

$$\begin{array}{lll}
t_2 & \Longrightarrow & < d_2, 1 >, \ \boldsymbol{< d_4, 1 >} \\
t_4 & \Longrightarrow & < d_2, 1 >, \ < d_3, 1 >, \ \boldsymbol{< d_4, 1 >}
\end{array}$$

  Using the inverted indexes we can see that d_4 has common terms with d_2 and d_3. Therefore, $S(d_4, d_2)$ and $S(d_4, d_3)$ similarity coefficients should be calculated.

- Consider d_5. Terms that appear in d_5: $\{t_3, t_5, t_6\}$. Their inverted indexes:

$$\begin{array}{lll}
t_3 & \Longrightarrow & < d_1, 1 >, \ < d_3, 1 >, \ \boldsymbol{< d_5, 1 >} \\
t_5 & \Longrightarrow & < d_1, 1 >, \ < d_2, 1 >, \ \boldsymbol{< d_5, 1 >}, \ < d_6, 1 > \\
t_6 & \Longrightarrow & \boldsymbol{< d_5, 1 >}, \ < d_6, 1 >
\end{array}$$

  Using the inverted indexes we can see that d_5 has common terms with d_1, d_2, d_3 and d_6. Therefore, $S(d_5, d_1)$, $S(d_5, d_2)$, $S(d_5, d_3)$ and $S(d_5, d_6)$ similarity coefficients should be calculated.

- Consider d_6. Terms that appear in d_6: $\{t_5, t_6\}$. Their inverted indexes:

$$\begin{array}{lll}
t_5 & \Longrightarrow & < d_1, 1 >, \ < d_2, 1 >, \ < d_5, 1 >, \ \boldsymbol{< d_6, 1 >} \\
t_6 & \Longrightarrow & < d_5, 1 >, \ \boldsymbol{< d_6, 1 >}
\end{array}$$

  Using the inverted indexes we can see that d_6 has common terms with d_1, d_2 and d_5. Therefore, $S(d_6, d_1)$, $S(d_6, d_2)$ and $S(d_6, d_5)$ similarity coefficients should be calculated.

The sum of the similarity coefficients that should be calculated is 22. Since, every similarity coefficient will have it's symmetric equivalent, we can calculate only the half of the similarity coefficients. Therefore, using the term inverted indexes approach we should calculate $\frac{22}{2} = 11$ document-by-document similarity coefficients.

(c)

Table 5 gives the length of each document.

Table 5: Lengths of the documents.

Document	Length
d_1	3
d_2	3
d_3	3
d_4	2
d_5	3
d_6	2

Let, S_{dice} be the similarity matrix that contains Dice coefficients and S_{cosine} be the similarity matrix that contains Cosine coefficients. We will calculate Dice coefficients using the formula $S_{dice}(d_i, d_j) = \frac{2*|d_i \cap d_j|}{|d_i|+|d_j|}$, and Cosine coefficients using the formula $S_{cosine}(d_i, d_j) = \frac{|d_i \cap d_j|}{|d_i|^{\frac{1}{2}}|d_j|^{\frac{1}{2}}}$. Both matrices, S_{dice} and S_{cosine} are shown below. Similarity coefficients that are different in two matrices are marked as bold.

$$S_{dice} = \begin{bmatrix} 1.0 & 0.333 & 0.667 & 0 & 0.667 & \mathbf{0.4} \\ - & 1.0 & 0.333 & \mathbf{0.8} & 0.333 & \mathbf{0.4} \\ - & - & 1.0 & \mathbf{0.4} & 0.333 & 0 \\ - & - & - & 1.0 & 0 & 0 \\ - & - & - & - & 1.0 & \mathbf{0.8} \\ - & - & - & - & - & 1.0 \end{bmatrix} \quad S_{cosine} = \begin{bmatrix} 1.0 & 0.333 & 0.667 & 0 & 0.667 & \mathbf{0.408} \\ - & 1.0 & 0.333 & \mathbf{0.816} & 0.333 & \mathbf{0.408} \\ - & - & 1.0 & \mathbf{0.408} & 0.333 & 0 \\ - & - & - & 1.0 & 0 & 0 \\ - & - & - & - & 1.0 & \mathbf{0.816} \\ - & - & - & - & - & 1.0 \end{bmatrix}$$

Kendall's tau measure is commonly used for comparing two separate rankings of the elements of a set. Basically, it gives the minimum number of element swaps needed to convert one ranking into another. Kendall's tau equals to 0 when both rankings are identical, because no element swap is required. Worst case occurs, when one ranking is in the reverse order of the other ranking. In that case Kendall's tau equals to $\frac{n*(n-1)}{2}$, because every possible pair should be swapped.

We consider our example. In our problem, for each document d_i there is a ranking of other documents - R_i. Here, R_i consists of all documents except d_i itself. Documents in R_i are ranked in decreasing order of their distance to d_i (closest documents are on the top, and farthest documents are on the bottom of the ranking). Two similarity measures (*dice* and *cosine*) may end up with two different rankings for some particular document d_i. One may use Kendall's tau rank correlation in order to see how are those two rankings differ from each other. In our particular example only the values marked in bold (matrices above) have changed. Fortunately, although those values have changed, the rankings of the documents for any d_i haven't changed. Therefore, we can say that Kendall's tau rank correlation equals to 0 for any d_i.

3 Question 3

In this part consider the paper J. Zobel, A. Moffat, "Inverted files for text search engines." *ACM Computing Surveys*, Vol. 38, No. 2, 2006.

a. Understand the skipping concept as applied to the inverted index construction.

Assume that we have the following posting list for term a: <1, 2> <3, 2> <9, 5> <10, 3> <12, 4> <17, 4> <18, 3>, <22, 2> <24, 4> <33, 4> <38, 5> <43, 4> <55, 3><64, 2> <68, 4> <72, 3> <75, 1> <88, 2>.. The posting list indicates that term-a appears in d1 twice and in d3 twice, etc.

Assume that we have the following posting list for term-b: <12, 2> <45, 2> <66, 1>.

Consider the following conjunctive Boolean query: term-a **and** term-b. If no skipping is used how many comparisons do you have to find the intersection of these two lists?

Introduce a skip structure, draw the corresponding figure then give the number of comparisons involved to process the same query.

State the advantages and disadvantages of large and small skips in the posting lists. Note that in the paper it is assumed that compression will be used. The skip idea is applicable in an uncompressed environment too.

b. Give a posting list of of term-a (above it is given in standard sorted by document number order) in the following forms: 1), a) ordered by $f_{d,t}$, b) ordered by frequency information in prefix form. What are the advantages of the approaches a and b? Do they have any practical value?

(a)

(i) Given two terms: *term-a* and *term-b*. Inverted document lists of the queries are as follows:

- *term-a* $\Longrightarrow < 1,2 >, < 3,2 >, < 9,5 >, < 10,3 >, < 12,4 >, < 17,4 >, < 18,3 >, < 22,2 >, < 24,4 >, < 33,4 >, < 38,5 >, < 43,4 >, < 55,3 >, < 64,2 >, < 68,4 >, < 72,3 >, < 75,1 >, < 88,2 >$
- *term-b* $\Longrightarrow < 12,2 >, < 45,2 >, < 66,1 >$

Given a Boolean query *term-a* **AND** *term-b*. In order to find the intersection of these two lists we should make **15 comparisons**. Both lists are sorted in increasing order of document identifiers. Hence, we intersect both lists in single scan over the longest list:

- Compare $< 12,2 >$ from *term-b*'s list with $< 1,2 >, < 3,2 >, < 9,5 >, < 10,3 >, < 12,4 >$ from *term-a*'s list. After **5** comparisons, we know where to insert the $< 12,2 >$ tuple.
- Compare $< 45,2 >$ from *term-b*'s list with $< 17,4 >, < 18,3 >, < 22,2 >, < 24,4 >, < 33,4 >, < 38,5 >, < 43,4 >, < 55,3 >$ from *term-a*'s list. After **8** comparisons, we know where to insert the $< 45,2 >$ tuple.
- Compare $< 66,1 >$ from *term-b*'s list with $< 64,2 >, < 68,4 >$ from *term-a*'s list. After **2** comparisons, we know where to insert the $< 66,1 >$ tuple.

(ii) Let's introduce skipping with chunk size = 4. There will be 5 chunks for *term-a*'s inverted list:

- *chunk-1* $\Longrightarrow < 1,2 >, < 3,2 >, < 9,5 >, < 10,3 >$. Chunk descriptor: $< 10,3 >$.
- *chunk-2* $\Longrightarrow < 12,4 >, < 17,4 >, < 18,3 >, < 22,2 >$. Chunk descriptor: $< 22,2 >$.
- *chunk-3* $\Longrightarrow < 24,4 >, < 33,4 >, < 38,5 >, < 43,4 >$. Chunk descriptor: $< 43,4 >$.

- *chunk-4* $\Longrightarrow < 55, 3 >, < 64, 2 >, < 68, 4 >, < 72, 3 >$. Chunk descriptor: $< 72, 3 >$.
- *chunk-5* $\Longrightarrow < 75, 1 >, < 88, 2 >$. Chunk descriptor: $< 88, 2 >$.

Let's follow the merge part step-by-step:

- **Merge $< 12, 2 >$: (3 comparisons needed)**
  - **Compare $< 12, 2 >$** with 1^{st} chunk's descriptor $\Longrightarrow$ continue.
  - **Compare $< 12, 2 >$** with 2^{nd} chunk's descriptor $\Longrightarrow$ it should be inserted into 2^{nd} chunk.
  - **Compare $< 12, 2 >$** with 1^{st} element of the 2^{nd} chunk $\Longrightarrow$ position found and merge completed.
- **Merge $< 45, 2 >$: (4 comparisons needed)**
  - We continue with the 2^{nd} chunk, where we stopped during last merge.
  - **Compare $< 45, 2 >$** with 2^{nd} chunk's descriptor $\Longrightarrow$ continue.
  - **Compare $< 45, 2 >$** with 3^{rd} chunk's descriptor $\Longrightarrow$ continue.
  - **Compare $< 45, 2 >$** with 4^{th} chunk's descriptor $\Longrightarrow$ it should be inserted into 4^{th} chunk.
  - **Compare $< 45, 2 >$** with 1^{st} element of the 4^{th} chunk $\Longrightarrow$ position found and merge completed.
- **Merge $< 66, 1 >$: (4 comparisons needed)**
  - We continue with the 4^{th} chunk, where we stopped during last merge.
  - **Compare $< 66, 1 >$** with 4^{th} chunk's descriptor $\Longrightarrow$ it should be inserted into 4^{th} chunk.
  - **Compare $< 66, 1 >$** with 1^{st}, 2^{nd} and 3^{rd} elements of the 4^{th} chunk $\Longrightarrow$ position found and merge completed.

Therefore, in order to find the intersection of the two lists using skipping with chunk size of 4, we should make **$(3 + 4 + 4) = 11$ comparisons**.

(iii) For large chunk sizes, the number of total chunks and the number of comparisons with chunk descriptors decreases (because there are less chunks), but the number of comparisons with document identifiers inside the selected chunk increases (because there are more documents in each chunk). On the other hand, for small chunks sizes the opposite conditions hold: the number of comparisons with document identifiers inside the selected chunk decreases (because there are less documents in each chunk), but the number of total chunks and the number of comparisons with chunk descriptors increases (because there are more chunks).

(b)

Inverted document list of *term-a* ordered by $f_{d,t}$ (*the frequency of term t in document d*):

$< 9, 5 >, < 38, 5 >, < 12, 4 >, < 17, 4 >, < 24, 4 >, < 33, 4 >, < 43, 4 >, < 68, 4 >, < 10, 3 >, < 18, 3 >, < 55, 3 >, < 72, 3 >, < 1, 2 >, < 3, 2 >, < 22, 2 >, < 64, 2 >, < 88, 2 >, < 75, 1 >$.

Inverted document list of *term-a* ordered by frequency information in prefix form:

$< 5 : 2 : 9, 38 >, < 4 : 6 : 12, 17, 24, 33, 43, 68 >, < 3 : 4 : 10, 18, 55, 72 >, < 2 : 5 : 1, 3, 22, 64, 88 >, < 1 : 1 : 75 >$.

4 Question 4

What are the components of an information retrieval test collection? Explain the pooling approach? Please read the paper by Zobel (How Reliable Are the Results of Large-Scale Information Retrieval Experiments?) and give some reflections of his criticism of this approach.

Components of a test collection in information retrieval are

- set of documents
- set of queries
- set of the systems

A system is another name for a methodology that ranks the documents for a given query. The ranking is usually based on the relevance of the document to the given query, i.e., the most relevant document is ranked as first and the least relevant document is ranked as last.

Pooling approach is used for identification of the relevant documents. For each query, top p documents from each run's rankings are collected in the pool (here, p is the *pool depth*). Document in the pool are assumed to be relevant documents for a given query. This entire method is called the *pooling approach*.

The article *"How Reliable are the Results of Large-Scale Information Retrieval Experiments?"* by Zobel brings few criticisms on pooling approach by giving experiment results on the data generated by TREC (Text Retrieval Conference). There are many experiments explained in the article.

First of all, Zobel writes the details of the experiment results that test the importance of pool depth. There are two disadvantages of pooling if pool size is not appropriate: system reinforcement and system omission. The former disadvantage may arise when the documents retrieved by two systems based on a similar idea can help each other and force the judgment to underestimate the effectiveness of the third system which is based on a completely different idea when compared with the ideas of the first two systems. On the other hand, latter case may occur for the queries with many answers. In this case, if only a small part of the relevant documents are added into pool, then the system which couldn't contribute the pool can be underestimated. This was explained in more detail, by giving experiment results where pool depth was decreased and each system's contribution was calculated by removing the retrieved documents from the pool, which were brought into pool by only that system. Both of the problems of pooling approach shows us that the pool depth is important.

Afterwards, Zobel says that pool depth may vary depending on the query. Article gives a pool depth estimation method that estimates the best pool depth for a given query and increases the number of retrieved relevant documents. This method incrementally increases the pool depth and predicts the best pool depth for a given query using the number of new relevant documents added into pool after the last depth increase.

5 Question 5

Please read the article entitled "Just the facts," by Marina Krakovsky *Communications of the ACM*, Vol. 56 No. 1, Pages 25-27, January 2013.

In that article there are two sides about news aggregation on the web. Please briefly summarize the arguments of each view. In your opinion which side of the aggregation debate is right and why? Please discuss briefly.

Krakovsky's article, *"Just the Facts"*, mentions the problems of how news aggregators may harm the original news sources. It emphasizes the defenses of both sides (content creators/owners and

aggregators) by giving quotations from many authorities including legal people, journalists and leading content aggregators.

According to Krakovsky, content creators complain that news aggregators take the facts from their original news, modify the original text and publish without any return to the news creators. An example complaint comes from a business journalist Robert Levine: "It's easy to make money by aggregating, investment is going to go to aggregators". This clearly states the threat to journalism and news creation industry.

Another concern mentioned in the article is the absence of legal protection of the content creators from being undermined by content aggregators. Even U.S. copyright law is not on the side of content creators. Federal laws conflict with state laws of few states. This is explained by Joseph Liu, a student of Intellectual Property and Internet Law at Boston College School of Law: "Are the media companies trying to protect something with state law that federal copyright law says you can't protect?"

On the other hand, content aggregators don't worry about these problems. While the president of Huffington Post Media Group (leading content aggregator), Arianna Huffington, defends themselves by saying that they give readers the most interesting stories from a broad range of news producers, actually Huffington Post rewrites the original text of the content using the facts given in the original content. Google News and Techmeme, another clear examples for content aggregation, give only the headline and couple of sentences, which encourages the reader to click on the original source of the content. Therefore, from content owners' point of view Google News and Techmeme seem to be a better news aggregators than Huffington Post.

In the end, article mentions the one year-old start-up, Newsright, that behaves as a platform between news creators and aggregators. Start-up aims to bring money from aggregators to the content owners.

Personally, I support content owners. Journalists collect facts and create news by investing a lot of hard-work into this. Sometimes they do their job in the middle of the war. From my point of view, it's unfair that aggregators, who just "copy" from the creators, may earn more money from the work that actually belongs to it's creator. As a solution, I suggest that Federal laws should protect the content creators by forcing the aggregators to pay money to creators for the actual work done behind the scenes. Otherwise, the size of the entire journalism industry will decrease by time, that will lead us into a huge trouble - *"journalism crisis"*.

6 Question 6

Please read the article entitled "Technology strategy and management: The Apple-Samsung lawsuits," by Michael A. Cusumano, *Communications of the ACM*, Vol. 56 No. 1, Pages 28-31, January 2013.

After reading this article please state your view on the impact of patents on innovations and social welfare.

The article *"Technology Strategy and Management: The Apple-Samsung Lawsuits"* by Cusumano summarizes how patents work in IT industry. Article gives a broad overview of the patent wars during the last century. It starts with Caddilac-Ford patent war example which occurred almost more than a century ago. Along with the last win of the Apple against Samsung in August 2012, Cusumano mentions other Apple-Samsung patent conflicts on Japanese and Korean courts that ended with Samsung wins. Cusumano also explains how companies reacted to the patents in the history, by giving Microsoft example.

Cusumano explains that one of the two possible scenarios may arise as a consequence of innovation protection. First is that patent protection may stimulate further innovation and push other companies to produce even better products and solutions. Second scenario is the sad one. Companies may just abandon trying to produce better products, and leave the entire market for dominating companies. When the history of big patent wars are analyzed carefully, one can easily see that innovation protection will never end up with the second scenario. As long as there is a gap for improvements, new companies will challenge bigger and market dominating companies. This is why I'm not against protecting innovative ideas. I think patents create a competitive environment for all parties. Without patents, most

probably innovation would still come only from big companies, but none of their competitor would challenge them with better innovation.

Another reason why I support patents is the following. Patents enable capitalism, by forcing other opponents to stop the progress. Unfortunately, capitalism is the only perfect system for technology improvements. That's why, patents along with the capitalist system pushes the technology forward. This is the fact that we see from the exponential growth in technology in last two decades. Therefore, I support patents.

7 Question 7

Please define the words lemma, root, and stem and give examples in English.

- **root** - smallest lexical unit of the word, that can't be divided into smaller constituents.
  The root of the word *technology* is **tech**.
- **stem** - part of the word that never gets affected, even when morphologically inflected. Stem is the base form of a word from which all inflected forms can be formed.
  The stem of the words *producer* and *production* is **produc**.
- **lemma** - base form or dictionary form of a word.
  The lemma of the words *write*, *written* and *wrote* is **write**.
  The lemma of the words *producer* and *production* is **produce**.
  The lemma of the words *go*, *going* and *went* is **go**.

8 Question 8

What is tf.idf? How would you use it in a binary D matrix environment. Please explain by using the D matrix given in the second question. In this question as a resouce please consider the Zobel-Moffat (2006) cited above.

TF·IDF is a *similarity measure* used for relevant document selection/retrieval for a given search query. To be more precise, search engines bring us r most relevant documents for a given search query string. This basically means, r most similar documents to the given search query string.

Zobel's survey *"Inverted Files for Text Search Engines"* gives a broad overview on search engine indexing. Search engines build their document indexes using the terms (words) that appear in documents, and when a search query comes (which is also composed of terms) search engine calculates the similarities between the indexed documents and the search query. Similarity is calculated by comparing a document with the query, based on their common terms (terms that appear in both of them). Few observations worth to spend time on:

- Term that appear in many documents should affect the similarity score with low weight.
- Term that appear in a document many times should affect the similarity score with high weight.
- Term that appear in a documents containing many terms (big document) should affect the similarity score with low weight.

Using the observations above one can use the similarity measure TF·IDF for document-query similarity calculations, where TF stands for *term frequency* and IDF stands for *inverted document frequency*. TF is high for the terms that appear in few documents, low for the terms that appear in many documents. IDF is high for the terms that appear many times in a document, and low for the terms that appear few times in a document. Multiplication of these 2 measures gives us the total similarity measure between a document and a search query.

Let's take the D matrix that was given in problem 2. Each of the terms 1, 2 and 6 appear in 2 documents. Term 3 and 4 appear in 3 documents. And, term 5 appears in 4 documents. Therefore, TF for term 5 will be the lowest, and TF for terms 1, 2 and 6 will be the highest.

Again from the same D matrix, we can see that the D matrix is a binary matrix (composed of only 0's and 1's). This means, that a term appears in some document only once or it doesn't appear at all. This brings us the same IDF values for all document-term pairs.

9 Question 9

What are the possible versions of caching in information retreival (again consider teh Zobel-Moffat paper). In your opinion which one is the most influential one in terms of efficiency, and in terms of effectiveness?

Let's start with the formal definition of the cache: "a component that stores data so that future requests for that data can be served faster". Few examples on cache: CPU cache, Disk cache, Web cache, etc. As it will be explain later, caching is one of the most crucial points of the modern search engines.

Zobel's survey on indexing in text search engines explains how the indexes are built, used, updated and maintained in modern search engines. Survey gives all details of the huge computational work done in less than a second for a single search query. For such expensive computations, it's important to cache the computed results and reuse them later. In other words, for large-scale search engines (Google, Bing) caching is a must.

Zobel mentions different types of information that can be cached:

- inverted lists can be kept in memory
- vocabulary can be kept in memory
- for phrase queries (containing few terms in some order), common term combinations can be cached
- the most important one is to cache the results of the common queries (the list of the most relevant r documents)

I agree with Zobel, that the most influential caching is the result caching.