CS533 HW1

Devrim Şahin

1. For Q1:

Rank	1	2	3	4	5	6	7	8	9	10
Relvnt		+		+		+				
Prec.	0	1/2	1/3	2/4	2/5	3/6	3/7	3/8	3/9	3/10
Recall	0	1/15	1/15	2/15	2/15	3/15	3/15	3/15	3/15	3/15

For Q2:

Rank	1	2	3	4	5	6	7	8	9	10
Relvnt	+		+		+				+	
Prec.	1/1	1/2	2/3	2/4	3/5	3/6	3/7	3/8	4/9	4/10
Recall	1/4	1/4	2/4	2/4	3/4	3/4	3/4	3/4	4/4	4/4

a. Then the interpolated precision tables are:

For Q1:

Recall	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
Prec.	0.50	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

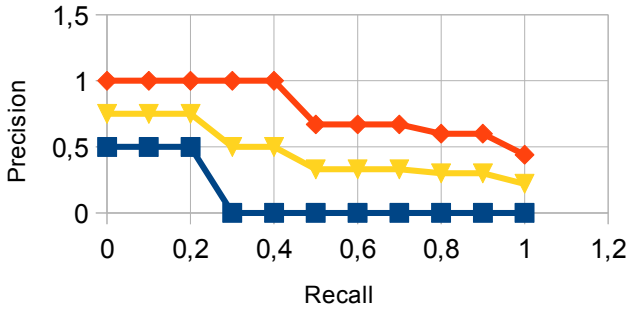
For Q2:

Recall	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
Prec.	1.00	1.00	1.00	1.00	1.00	0.67	0.67	0.67	0.60	0.60	0.44

Average:

Recall	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
Prec.	0.75	0.75	0.75	0.50	0.50	0.33	0.33	0.33	0.30	0.30	0.22

Below is the plot. Blue is Q1, Orange is Q2. Yellow is the average:



b. Q1: 15 relevant documents. 15 results were never returned, but we can extrapolate and say that for 15 results, 3 relevant documents were obtained. Therefore R-precision is 3/15 = **0.2**

Q2: 4 relevant docs. in total; 2 relevant documents in the first 4 results; R-prec = 2/4 = **0.5**

Average R-prec: (0.2 + 0.5)/2 = **0.35**

c. MAP for Q1 = (0.5 + 0.5 + 0.5)/15 = **0.10**
MAP for Q2 = (1 + 2/3 + 3/5 + 4/9)/4 = **0.67**
Avg. = **0.38**

2. Assume we have 2 machines, M_1 and M_2. In a possible document-based partitioning scheme d_1, d_2 and d_3 will be stored in M_1 whereas d_4, d_5 and d_6 will be sent to M_2. Whereas, in term-based partitioning, we will make the assignment of terms instead, such that $t_{1..3}$ -> M_1 and $t_{4..6}$ -> M_2.

Or, we can partition them more cleverly, easing the search process; such that $d_{1,5,6}$ -> M_1 and vice versa. This way we can assert that all terms in M_1 have t_6, whereas none in M_2 has. A similar thing can be done in term-based partitioning for d_1 by assigning $t_{1,5,6}$ to M_1.

In cases where this is not possible (which generally is the case), one should distribute the query to all machines and finally merge results. The inverted index constructed on terms for the document-based partitioning scheme will require looking for t_i in both M_1 and M_2, then merging the list of documents. For the term-based partitioning scheme, we should find the machine in which t_i is stored and get the entire list of documents.

According to Zobel et al., document-based partitioning typically results in a better balance of workload than does term partitioning and achieves superior query throughput. Furthermore, term-based partitioning requires the exchanging of index fragments, which makes it immediately noticeable when one of the machines is down. Therefore document-based partitioning seems more plausible.

3. **a.**

term a: <1, 5> <4, 1> <9, 3> <10, 4> <12, 4> <17, 4> <18, 3>
<22, 2> <24, 4> <33, 4> <38, 5> <43, 5> <55, 3><64, 2>
<68, 4> <72, 5> <75, 1> <88, 2>
term b: <12, 3> <40, 2> <66, 1>

Constructing chunks of 4 entries with descending order w.r.t. the document number:

term a:
chunk1: <10, 4> <9, 3> <4, 1> <1, 5>
chunk2: <18, 3> <17,4> <12, 4> <12, 4>
chunk3: <38, 5> <33, 4> <24, 4> <22, 2>
chunk4: <68, 4> <64, 2> <55, 3> <43,5>
chunk5: <88, 2> <75, 1> <72, 5>
term b:
chunk1: <66, 1> <40, 2> <12, 3>

The chunking scheme is as follows: Given a document d_i, we first check chunk1. For instance, assume we are looking for term a. Checking chunk1, if $d_i > 10$, we can skip the chunk and advance to chunk2. Otherwise, we should sequentially check chunk1 and stop there (Note that if d_i=10 we should also check chunk2, because of possible repetitions). We apply the same pruning approach to every chunk using the first term in the chunk. Given that there are 19 documents containing term a, and 3 terms containing term b; without a skipping strategy we would make 19*3 = **57** comparisons for "term a AND term b" query. With the given skipping strategy, however, we can start from term-b and do this:

for 66:

compare chunk1: 66>10 skip
compare chunk2: 66>18 skip
compare chunk3: 66>38 skip

compare chunk4: 66<68 check
compare 66!=64, also 66>64 therefore break
skip chunk5 because all elements in it are greater than 68 anyway,
and we are looking for 66

(5 comparisons with no matches)

Similarly;

for 40: 7 comparisons with no matches
for 12: 5 comparisons with 2 matches

In total we make **17** comparisons with skipping, and **57** comparisons without.

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

2. <1:2:4,75> <2:3:22,64,88> <3:3:9,18,55> <4:6:10,12,17,24,33,68> <5:4:1,38,43,72>
And if we take differences as well for further compression:
<1:2:4,71> <2:3:22,42,24> <3:3:9,9,37> <4:6:10,2,5,7,9,35> <5:4:1,37,5,29>

Ordering terms by frequency increases performance because one can disregard terms with frequencies below a level, greatly reducing disk I/O. Prefix construction is costy, but allows further compression by avoiding repetition of frequency values.

4. In Cranfield approach to IR experimens, we treat the evaluation problem as if we work in a laboratory environment. That is, we have a test collection consisting of a set of documents, a set of queries, and relevant documents for each query. One example to this is the TREC test collection. To determine relevant documents for each query, one can devise a pooling approach, in which a set of assessers (or 'annotators') deem the top results retrieved by several information retrieval systems for a given query relevant or irrelevant. The union of the relevant document set is then picked as the ground truth for future experiments.

5. The empirical investigations that were conducted by Justin Zobel in the relevant paper show that TREC results that are obtained through the pooling approach are reliable in terms of measured relative performance; but fails to meet demands in terms of recall; that is, many relevant documents were amiss in the merged result sets.

bpref is the binary preference based evaluation measure that considers whether relevant documents are ranked above irrelevant ones. bpref can be used once relevant and irrelevant items are defined, that is, a relevance judgement must be passed on a pool of IR systems by assessers, as defined above.

6. **a.** Unlike traditional IR environments, a data stream is potentially unbounded; that is, there might be massive amount of data arriving sequentially, whereas in the traditional IR environments there exists a finite dataset in which the instances can be processed in an order defined by the algorithm rather than the stream itself. Furthermore, traditional IR environments have fairly static probability distributions, although that is not necessarily the case for a data stream.

b. A time window is a frame in which recent instances are stored. The definition of 'recency' here defines the characteristics of the time window. For example we can define a time window that stores the last 10 elements. Similarly, we can limit our time window with respect to the timestamp of the packages that arrive and say that we want to keep the instances that arrived in the last 3 seconds. One of the advantages of keeping a time window is that it allows our operator to be

stateful. Also, under certain assumptions, one can treat the dataset inside a time window as if it was a traditional IR dataset.

c. Data abstraction in this concept refers to on-line summarization of the data that is input to the stream such that the produced summary represents the necessary information for the clustering system without having the satellite data.

d. Feature vectors: Also named as the Clustering Feature (CF) vector, it has 3 components:

N: the number of data objects
LS: the linear sum of data objects (n-dim array)
SS: the squared sum of data objects (n-dim array)

These are selected for being common terms in the equations calculating cluster centroids, radii and diameters:

$$centroid = \frac{LS}{N}$$

$$radius = \sqrt{\left(\frac{SS}{N} - \left(\frac{LS}{N}\right)^2\right)}$$

$$diameter = \sqrt{\left(\frac{2N * SS - 2 * LS^2}{N(N-1)}\right)}$$

e. Since the data abstraction step provides summary statistics, of which the size is much less than the streaming data; one can employ offline algorithms such as k-means to work on the streaming data that is chunked into windows.

8. **a.** Nominal data is a form of discrete data where possible values are finite and unordered. That is, the user chooses one of the options, of which the order does not mean anything. An example is gender: M/F (this is also called *dichotomous*, for it has only two categories). Another example is "your favourite car brand". There are also Ordinal, Interval and Ratio data. Below is a chart depicting their features:

Provides:	Nominal	Ordinal	Interval	Ratio
"Counts," aka "Frequency of Distribution"	✔	✔	✔	✔
Mode, Median		✔	✔	✔
The "order" of values is known		✔	✔	✔
Can quantify the difference between each value			✔	✔
Can add or subtract values			✔	✔
Can multiple and divide values				✔
Has "true zero"				✔

b. The data in a binary D matrix is *ordinal*, for the set of terms are discrete and their order is important.

c. We cannot, because we do not know how similar each of the nominal values to each other. Bin Wang suggested in his paper "A New Clustering Algorithm On Nominal Data Sets" to use 'Olary code' (a binary sequence which has k+1 possible values given k bits) for assigning numeric values to nominal data. Then using this Olary transform, he applies an algorithm similar to k-means, however uses the Hamming distance between two Olary codes instead of the Euclidean distance.