COMPUTER ENGINEERING DEPARTMENT
BILKENT UNIVERSITY

Assignment No. 1

Abdurrahman Yasar

June 10, 2014

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, 15 and 4 (in Q1 12 of the relevant documents are not retrieved).

1.1 A. TREC INTERPOLATION RULE

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).

TREC Interpolation Rule: 'Interpolated' means that, for example, precision at recall 0.10 (i.e., after 10% of rel docs for a query have been retrieved) is taken to be MAXIMUM of precision at all recall points>= 0.10. Values are averaged over all queries (for each of the 11 recall levels). These values are used for Recall-Precision graphs.

$$Recall = \frac{number\ of\ relevant\ items\ retrieved}{number\ of\ relevent\ items\ in\ collection}$$

$$Precision = \frac{number\ of relevant\ items\ retrieved}{total\ number\ of\ items\ retrieved}$$

Rank	1	2	3	4	5	6	7	8	9	10
Recall	0	0.067	0.067	0.133	0.133	0.200	0.200	0.200	0.200	0.200
Precision	0	1/2	1/3	2/4	2/5	3/6	3/7	3/8	3/9	3/10

Table 1.1: Recall- Precision Table For Q1

Recall	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Precision	1/2	2/4	3/6	0	0	0	0	0	0	0	0

Table 1.2: Interpolated Recall- Precision Table For Q1

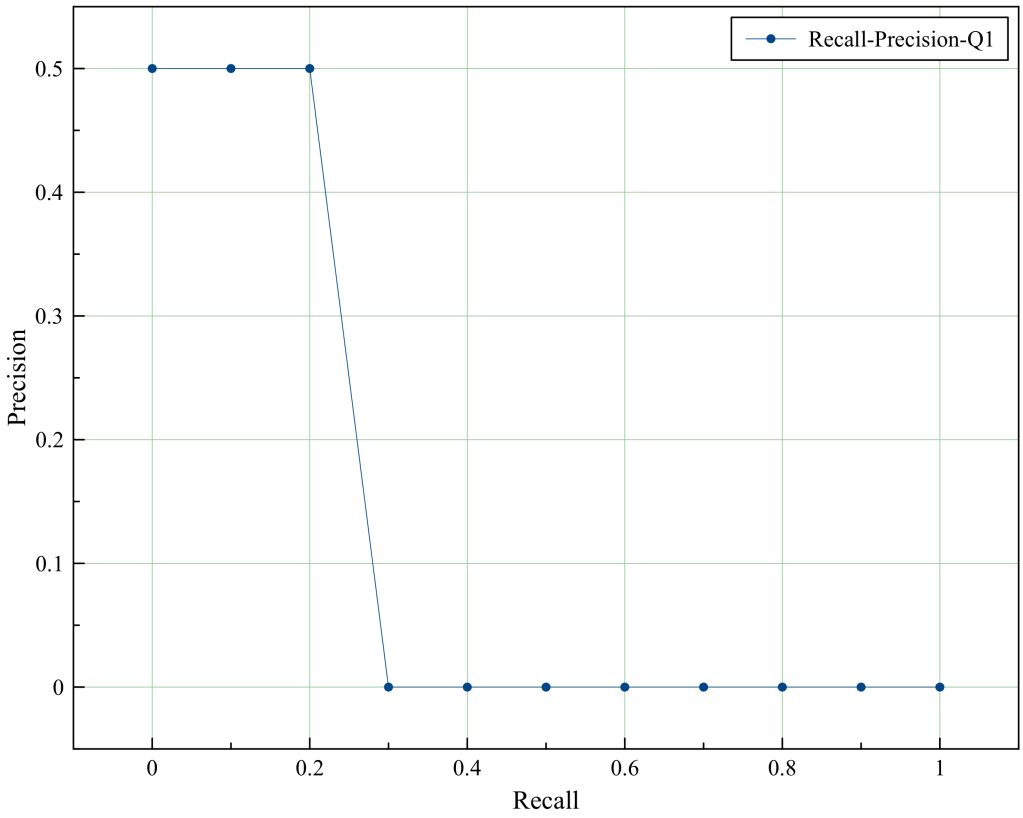


Figure 1.1: Recall-Precision Graph of Q1

Rank	1	2	3	4	5	6	7	8	9	10
Recall	0.25	0.25	0.5	0.5	0.75	0.75	0.75	0.75	1.0	1.0
Precision	1	1/2	2/3	2/4	3/5	3/6	3/7	3/8	4/9	4/10

Table 1.3: Recall- Precision Table For Q2

Recall	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Precision	1	1	1	2/3	2/3	2/3	3/5	3/5	4/9	4/9	4/10

Table 1.4: Interpolated Recall- Precision Table For Q2

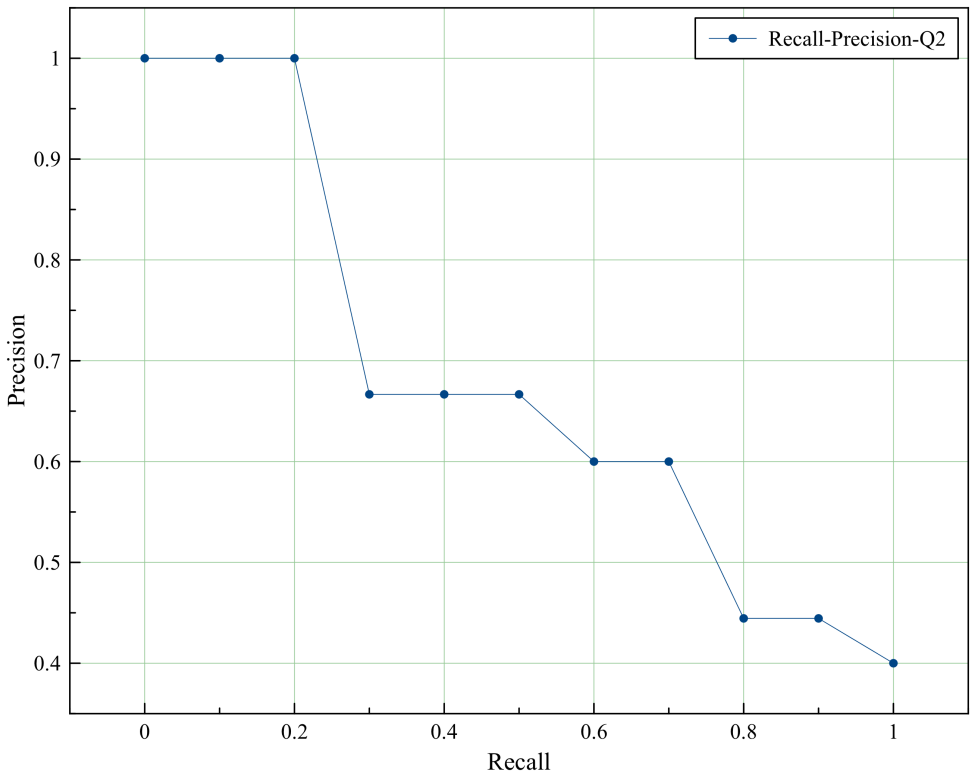


Figure 1.2: Recall-Precision Graph of Q2

1.2 B. R-PRECISION

Find R-Precision (TREC-6 AppendixA for definition) for Query1 and Query2.

R-Precision: R-Precision is the precision after R documents have been retrieved where R is the number of relevant documents for the topic. So in our case:

- Q1: We cannat calculate R-Precision for Q1 because the number of relevant documents for Q1 is 15 so R is 15 but we have only retrived 10 documents so we cannot say anything about the last 5.
- Q2: R is 4 and there are 2 relevant documents in 4 retrieved documents so **R-Precision** $= \frac{1}{2}$

- For Q1 we have 0 becaus we couldn't calculate its R-Precision and for Q2 we have 0.5. Then **Avarage :** $\frac{0+\frac{1}{2}}{2} = \frac{1}{4}$

1.3 C.FIND MAP

Find MAP for these queries. If it is not possible to calculate explain why.

- Average precision of Q1 : $\frac{0.067+0.133+0.200+12*0.000}{15} = 0.027$
- Average precision of Q1 : $\frac{1+\frac{2}{3}+\frac{3}{5}+\frac{4}{9}}{4} = 0.677$
- MAP : $\frac{0.027+0.677}{2} = 0.352$

2 QUESTION 2

Consider document-based partitioning and term-based partitioning approaches as define in the Zobel-Moffat paper "Inverted files for text search engines" (ACM Computing Surveys, 2006). Please also consider the following document by term binary D matrix for m= 6 documents (rows), n= 6 terms (columns). Describe a two ways of indexing across a cluster of machines.

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

Document Based Partitioning: The simplest distribution regime is to partition the collection and allocate one subcollection to each of the processors.

Term Based Partitioning: In a term-partitioned index, the index is split into components by partitioning the vocabulary.

Lets give an example partitioning assuming the above D matrix. We have 6 documents; d_1, d_2,..., d_6 , 6 terms t_1, t_2,..., t_6 and assume that we have 3 computers; c_1, c_2, c_3. If we use document based partitioning 2 document will be sent to a computer in other words we send rows to computers:

$$c_1 \rightarrow \{d_1, d_2\}$$
$$c_2 \rightarrow \{d_3, d_4\}$$
$$c_3 \rightarrow \{d_5, d_6\}$$

If we use term based partitioning we send same terms of the documents, in other words columns, to computers:

$$c_1 \rightarrow \{t_1,\ t_2\}$$
$$c_2 \rightarrow \{t_3,\ t_4\}$$
$$c_3 \rightarrow \{t_5,\ t_6\}$$

These partitioning methods is important for balancing the workload. For example in the above example using document based partitioning is better because each document has at least 2 and at most 3 terms so the workload of computer will be relatively equal to each other. But if we use term-based partitioning we see that for each term appears at least 2 at most 4 in documents so this type of partitioning gives us a worse load balancing.

In conclusion after looking distribution of terms in documents and number of terms per document we can decide one of the partitioning methods which is more uniformly distributed to obtain a good workload balancing.

3 QUESTION 3

In this part again consider the Zobel-Moffat paper.

3.1 A. SKIPPING CONCEPT

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

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

Assume that we have the following posting list for term - b: $<12,3><40,2><66,1>$

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.

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

Boolean query: term-a **and** term-b. Given these two inverted lists term-a and term-b we can find intersection of the above lists without using skipping in **15 comparisons**. Since the lists are in the order of increasing document identifiers we can intersect them in one scan.

- Compare $< 12,3 >$ of term-b's list with $< 1,5 > < 4,1 > < 9,3 > < 10,4 > < 12,4 >$ from term-a's list. After **5 comparisons** then we know where to place $< 12,3 >$.
- Compare $< 40,2 >$ of term-b's list with $< 17,4 > < 18,3 > < 22,2 > < 24,4 > < 33,4 > < 38,5 > < 43,5 >$ from term-a's list. After **7 comparisons** then we know where to place $< 40,2 >$.
- Compare $< 66,1 >$ of term-b's list with $< 55,3 > < 64,2 > < 68,4 >$ from term-a's list. After **3 comparisons** then we know where to place $< 66,1 >$.

I wil use skipping with chunk size = 4. To do this we will split long list to chunks; in our case this is the list of term-a. So, there will be 5 chunks for term-a's inverted list.

- **chunk-1:** $\rightarrow < 1,5 > < 4,1 > < 9,3 > < 10,4 >$ chunk descriptor: $< 10,4 >$
- **chunk-2:** $\rightarrow < 12,4 > < 17,4 > < 18,3 > < 22,2 >$ chunk descriptor: $< 22,2 >$
- **chunk-3:** $\rightarrow < 24,4 > < 33,4 > < 38,5 > < 43,5 >$ chunk descriptor: $< 43,5 >$
- **chunk-4:** $\rightarrow < 55,3 > < 64,2 > < 68,4 > < 72,5 >$ chunk descriptor: $< 72,5 >$
- **chunk-5:** $\rightarrow < 75,1 > < 88,2 >$ chunk descriptor: $< 88,2 >$

Here you see the comparisons for skipping concept:

- **Merge** $< 12,3 >$
  1. Compare with first chunk's descriptor. $10 < 12$ so go to next chunk
  2. Compare with second chunk's descriptor $12 < 22$ so we will insert $< 12,3 >$ into this chunk
  3. Compare with <12,4>. We have found the position so this merge is completed.
- **Merge** $< 40,2 >$
  1. Compare with first chunk's descriptor. $10 < 40$ so go to next chunk
  2. Compare with second chunk's descriptor. $22 < 40$ so go to next chunk
  3. Compare with third chunk's descriptor $40 < 43$ so we will insert $< 40,2 >$ into this chunk
  4. Compare with $< 24,4 >$. $24 < 40$ so go to next tuple.
  5. Compare with $< 33,4 >$. $33 < 40$ so go to next tuple.
  6. Compare with $< 38,5 >$. $38 < 40$ so go to next tuple.
  7. Compare with $< 43,5 >$. We have found the position so this merge is completed.
- **Merge** $< 66,1 >$
  1. Compare with first chunk's descriptor. $10 < 66$ so go to next chunk

2. Compare with second chunk's descriptor. 22 < 66 so go to next chunk
3. Compare with third chunk's descriptor. 43 < 66 so go to next chunk
4. Compare with fourth chunk's descriptor 66 < 72 so we will insert < 66, 1 > into this chunk
5. Compare with < 55, 3 >. 55 < 66 so go to next tuple.
6. Compare with < 64, 2 >. 64 < 66 so go to next tuple.
7. Compare with < 68, 4 >. We have found the position so this merge is completed.

Using skipping method we have made **17 comparisons.**

3.1.1 B. Posting List

Give a posting list of term-a (above it is given in standard sorted by document number order) in the following forms: **a)** ordered by $f_{d,t}$, **b)** ordered by frequency information in prefix form.

- Ordered by $f_{d,t}$ (the frequency of term t in document d): , < 1,5 >, < 38,5 >, < 43,5 > , < 72,5 >, < 10,4 >, < 12,4 >, < 17,4 >, < 24,4 >, < 33,4 >, < 68,4 >, < 9,3 >, < 18,3 > , < 55,3 >, < 22,2 >, < 64,2 >, < 88,2 >, < 4,1 >, < 75,1 >
- Ordered by frequency information in prefix form: < 5 : 4 : 1, 38, 43, 72 >, < 4 : 6 : 10, 12, 17, 24, 33, 68 >, < 3 : 3 : 9, 18, 55 >, < 2 : 3 : 22, 64, 88 >, < 1 : 2 : 4, 75 >

List with frequency order could improve the query processing time. We can look ter frequency of documents and the document with same terms with high frequencies could be considered nearly the same and the ones that is very different with term frequencies could be considered as irrelevant. To make this more significant I prefer to use a post list ordered by frequency information in prefix form. Because using such a list doing comparisons are more efficient because there is no need to process entire list.

4 Question 4

What is meant by Cranfield approach to testing in information retrieval?

The Cranfield approach uses test collections to evaluate documents and information retrieval: standardised resources used to evaluate information retrieval systems with respect to system. The main components of an information retrieval test collection are the document collection, topics, and relevance assessments. These, together with evaluation measures, simulate the users of a search system in an operational setting and enable the effectiveness of an information retrieval system to be quantified. Evaluating information retrieval systems in this manner enables the comparison of different search algorithms and the effects on altering algorithm parameters to be systematically observed and quantified.

The most common way of using the Cranfield approach is to compare various retrieval strategies or systems, which is referred to as comparative evaluation. In this case the focus is on the relative performance between systems, rather than absolute scores of system effectiveness. To evaluate using the Cranfield approach typically requires these stages: (1) select different retrieval strategies or systems to compare; (2) use these to produce ranked lists of documents (often called runs) for each query (often called topics); (3) compute the effectiveness of each strategy for every query in the test collection as a function of relevant documents retrieved; (4) average the scores over all queries to compute overall effectiveness of the strategy or system; (5) use the scores to rank the strategies/systems relative to each other.

5 QUESTION 5

Is pooling a reliable approach for the construction of test collections ? Find the paper by Zobel regarding this issue, it may help. What is bpref? Is it anyway related to the pooling concept?

The pooling method examines the top ranked k documents from each n independent retrieval efforts. If k and n are large, the set of documents judged relevant may be assumed to be representative of the ideal set therfore suitable for evaluating retrieval results.

One of the disadvantages of having measurement depth m exceed pool depth p is that similar systems can reinforce each other. Consider a pool of three systems, A, B, and C. Suppose A and B use similar retrieval mechanisms such that some of the documents retrieved by A at a depth between p and m are retrieved by B at depth less than p ,and vice versa; and suppose C is based on different principles to A and B. Then the performance of C can be underestimated. That is, this methodology may misjudge the performance of novel retrieval techniques.

Another potential disadvantage of pooling is that, there could be a technique whose effectiveness is underestimated because of inopportunity to contribute the pool which is caused by identifying only a fraction of the relevant documents.

6 QUESTION 6

Please study the paper "Data stream clustering," by Silva et al. ACM Computing Survey, 2013.

6.1 A. DATA STREAM VS. TRADITIONAL I.R. ENVIRONMENTS

Describe the similarities and differences of the data stream and traditional information retrieval environments.

- Traditional information retrieval systems run querries on a collection of information which is available.

- In a data stream manner data flows so the systems make filtering the information; in other words the data is not static.
- In traditional information retrieval first we retrieve documents then process them in data stream we retrieve and process documents at the same time.
- In traditional data clustering is more accurrate because we already know the data in data stream we don't

6.2 B. WINDOW MODEL

What is the concept of time window, what are the advantages of using time window?

In a data streaming system a stream could change the data distribution. Systems who gives the equal importance to new and old data cannot capture these changes. To avoid that kind of problems window models have been proposed. In this type of models you have a window where you store incomming streams and then process them. This storing and storing and processing steps depends on the time window model that you use. Actually there are 3 common types of window models which are; (i) sliding windows, (ii) damped windows and (iii) landmark windows. All these models try to efficiently use incomming data and avoid the problems like we described in the beginning of the paragraph.

6.3 C. ABSTRACTION

What is meant by abstraction?

Here by abstraction they mean summarizing the data to deal with space and memory constraints of stream applications. With summarizing the stream and preserve their meaning without the need of actually storing them.

6.4 D. DATA STRUCTURES

Choose a data structure defined in the data abstraction part and explain its components?

Prototype Array: Briefly a prototype array is a simplified summarization data structure. It summarizes the data partition and stores prototypes like medoids and centroids.

For examle in Stream [Guha et al. 2000] there is an array of prototype. To summarize the stream they divide it into chunks. Each chunk summarized in in 2k representative objects by using a variant of k-medoids algorithm. Compressing the descriptions is repeated until an array of m prototypes is obtained. Next these m prototypes are further compressed into 2k prototypes and the process continues along the stream.(See fig. 6.1)

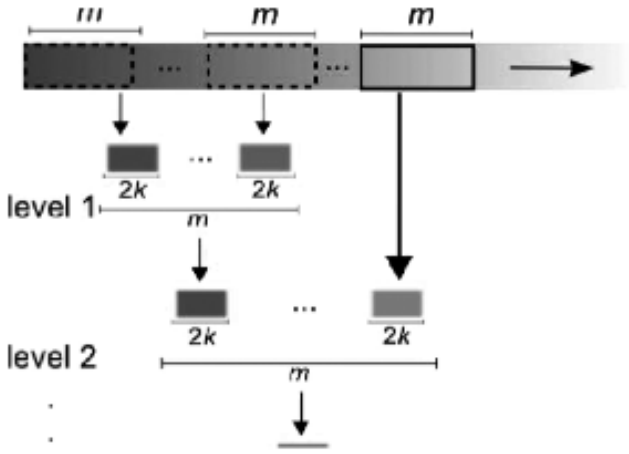


Figure 6.1: Overviewof Stream [Guha et al. 2000], which makes use of a prototype array

6.5 E. CLUSTERING

In data stream clustering how do we use the standard clustering algorithms such as k-means?

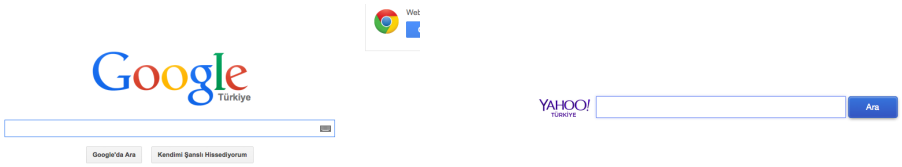
One of the most effective idea to cluster streams is usage of CF vectors. To adopt CF vectors to k-means clustering there are 3 ways:

- Calculate the centroid of each CF vector. Each centroid will be clustered by k-means
- Same as the previous item but at this time we use weighting.
- Aooley the clustering algorithm directly to the CF vectors since thier components keep the sufficient statistics for calculating mos of the required distances and quality metrics.

7 QUESTION 7

Study the paper "A survey of Web clustering search engines" by Carpineto et al. ACM Computing Survey, 2009. Find two of the web search engines defined in the paper and compare their user interface, performance in general, by conduction a simple set experiments. Please also define your experiments. If the eng ines mentioned in the paper do not exist find other similar search engines on the web. Note that you are not writing a research you are trying to convince people with IR knowledge. Try to be more convincing by providing some quantitative observations.

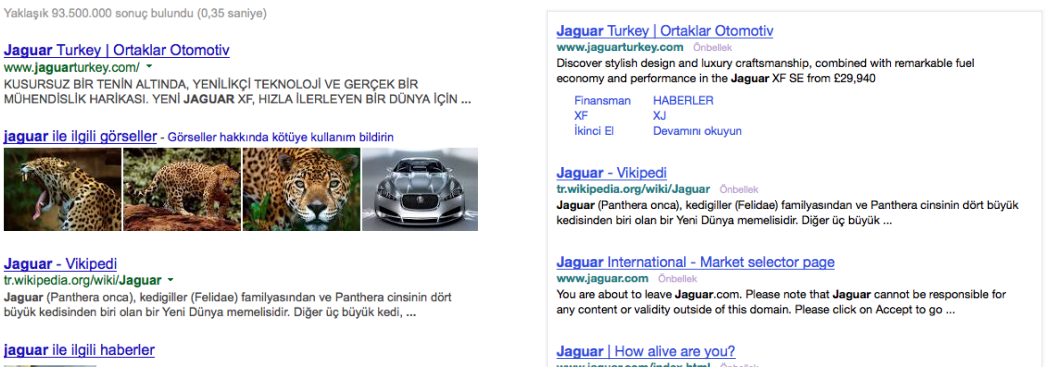
For this question I chose to compare Google's and Yahoo's search engines. Let's begin with **user interface**; as you can see in the following figures their user interfaces are almost the same. It has one text box and a serch button.



(a) Google Search Engine (b) Yahoo Search Engine

Figure 7.1: User Interfaces

To compare the results I have made 30 tests; 10 for just words like jaguar, ankara, paris, bosphor etc. 10 for searches with 2 to 5 words and 10 for searches 6 to 10 words. After these tests with single word searches the top ten results of google and yahoo is almost the same but when we increase the number of words the accuracy between them decreases. In my tests I have obtained 95% of accuracy for 1 word search, 78% accuracy for 2 to 5 word searches and 63% accuracy for 6 to 10 word searches. But also instead of looking top ten if we look top 15 results these persentages increases a little.



(a) Google Search for Jaguar (b) Yahoo Search for Jaguar

Figure 7.2: Search Result Example

After my tests I can say that top 3-4 results of google are better than yahoo. For example when I search something that is relevant to a research topic google directly returns some papers about it but yahoo don't. So if you want to find more relevant pages for your searching phrase in the top 3-4 of the list google is better.

8 QUESTION 8

On data forms used in clustering:

8.1 A. Nominal Data

What is nominal data? Give two examples

Nominal data are items which are differentiated by a simple naming system. For example:

- Asian countries
- The number pinned on a building

8.2 B. D Matrix

What is the type of data (D matrix) we use in document clustering (nominal etc.)?

D matrix is in the ratio type of data because $D^{th}_{i,j}$ element of D matrix gives us the number of occurances of term j in document i.

8.3 C. Nominal Data in Clustering

Can we use nominal data in clustering algorithm we have in IR? If your answer is no how can we convert nominal data into a meaningful form so that they can be used for clustering? Do some research on this on the Web. State your references

Actually we cannot use nominal data types in clustering algorithms we have in IR. Because in this type of clustering algorithms we need a distance measure. So at this point to use nominal data for clustering firstly we need to normlize them to a form that we can calculate distances. We cannot do this with appointing random integers to these data because similarities must be conserved. After this transformation then we can use our clustering algorithms. In the web we see several papers that proposes some transformation methods:

1. A New Clustering Algorithm On Nominal Data Sets, Bin Wang
2. Clustering Categorical Data, Steven X. Wang
3. Clustering nominal and numerical data: a new distance concept for an hybrig genetic algorithm