Old Dominion University
Department of Computer Science
CS834: Introduction to Information Retrieval
Fall 2017
Assignment 4
Professor: Dr. Michael Nelson

Hussam Hallak
CS Master's Student

November 30, 2017

Question 1:

Exercise 8.3:

For one query in the CACM collection (provided at the book website), generate a ranking using Galago, and then calculate average precision, NDCG at 5 and 10, precision at 10, and the reciprocal rank by hand.

Answer:

I first built the index for the CACM collection using the "build" command in Galago.

```
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago build --indexPath=/home/hussam/Desktop/galago/galago-bin/bin/index --inputPath=/home/hussam/Desktop/galago/galago-bin/bin/cacm --filetype=html
```

I decided to run query #12 "portable operating systems".

I created the file named "83.json" and saved the query in it as directed by Galago "batch-search" command help.

```
{
    "queries" : [
    {
        "number" : "CACM-12",
        "text" : "#combine(portable operating systems)"
    }
    ]
  }
```

Listing 1: The content of 83.json

I generated the ranking list using Galago from the command line:

```
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago batch-search --index=/home/hussam/Desktop/galago/galago-bin/bin/index --requested=10 /home/hussam/Desktop/galago/galago-bin/bin/queries/83.json
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-3127.html 1 -5.64732220 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1930.html 2 -6.30066873 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2246.html 3 -6.35308840 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-3196.html 4 -6.44900997 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2593.html 5 -6.68985970 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1591.html 6 -6.86473945 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1680.html 7 -6.87638212 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2740.html 8 -6.91784238 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-3068.html 9 -6.95520083 galago
CACM-12 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2319.html 10 -6.97755390 galago
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$
```

To answer this question, I need to calculate the following values by hand:

1. Average precision
2. NDCG at 5
3. NDCG at 10
4. Precision at 10
5. Reciprocal rank

To get the average precision, we need to compute the precision for each relevant document that is retrieved, then take the average.

Precision = Number of relevant documents/Number of retrieved documents

When calculating the average precision, we only consider relevant documents.

I downloaded the relevance judgment file "cacm.rel". The section for query 12 is:

```
12 Q0 CACM-1523 1
12 Q0 CACM-2080 1
12 Q0 CACM-2246 1
12 Q0 CACM-2629 1
12 Q0 CACM-3127 1
```

Therefore the number of relevant documents in the collection is 5. Each document in the result set can be identified as a "Hit" or "Miss":

Result	File Name	Hit/Miss
1	CACM-3127.html	Hit
2	CACM-1930.html	Miss
3	CACM-2246.html	Hit
4	CACM-3196.html	Miss
5	CACM-2593.html	Miss
6	CACM-1591.html	Miss
7	CACM-1680.html	Miss
8	CACM-2740.html	Miss
9	CACM-3068.html	Miss
10	CACM-2319.html	Miss

The average precision can be calculated:
Average Precision = (1+2/3)/2 = 0.83
The reason that the average precision did not change is because there are no relevant documents returned after the third position. Again, When calculating average precision, only relevant documents are considered in the calculation.

NDCG: The approach for calculating NDCG can be summarized in the following steps:

1. Set a relevance level for each document. I used 1 if the document is relevant and 0 otherwise.

2. Calculate the DCG using this formula:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i}$$

   where p is the rank, i is the rank for each document, and rel_i is the relevance level for each document.

3. Create perfect ranking (simpler with binary).

4. Calculate the ideal DCG

5. Calculate NDCG using the formula:

$$NDCG_p = \frac{ActualDCG_p}{IdealDCG_p}$$

For our result set of 10 documents judged on a binary scale:
1, 0, 1, 0, 0, 0, 0, 0, 0, 0
Discounted gain:
1, 0, 1/1.59, 0, 0, 0, 0, 0, 0, 0 = 1, 0, 0.63, 0, 0, 0, 0, 0, 0, 0
Actual DCG values:

1, 1, 1.63, 1.63, 1.63, 1.63, 1.63, 1.63, 1.63, 1.63
DCG at rank 5 = 1.63
DCG at rank 10 = 1.63
Perfect Ranking:
1, 1, 0, 0, 0, 0, 0, 0, 0, 0
Discounted gain (for perfect ranking):
1, 1, 0, 0, 0, 0, 0, 0, 0, 0
Ideal DCG values:
1, 2, 2, 2, 2, 2, 2, 2, 2, 2
NDCG = Actual DCG / Ideal DCG
NDCG:
1, 0.5, 0.82, 0.82, 0.82, 0.82, 0.82, 0.82, 0.82, 0.82
NDCG at rank 5 = 0.82
NDCG at rank 10 = 0.82
Precision at 10:
For our result set of 10 documents judged on a binary scale:
1, 0, 1, 0, 0, 0, 0, 0, 0, 0
Precision at each rank:
1, 0.5, 0.67, 0.5, 0.4, 0.33, 0.29, 0.25, 0.22, 0.2
Precision at 10 = 0.2

The reciprocal rank is the reciprocal of the rank at which the first relevant document is retrieved. Since, the first relevant document retrieved at rank 1, the reciprocal rank is $1/1 = 1$

Question 2:

Exercise 8.4:

For two queries in the CACM collection, generate two uninterpolated recall precision graphs, a table of interpolated precision values at standard recall levels, and the average interpolated recall-precision graph.

Answer:

I decided to run query #13 "code optimization for space efficiency" and query #27 "memory management aspects of perating systems".

I created the file named "84.json" and saved the queries in it as directed by Galago "batch-search" command help.

```
{
    "queries" : [
    {
        "number" : "CACM-13",
        "text" : "#combine(code optimization for space efficiency)"
    },
    {
        "number" : "CACM-27",
        "text" : "#combine(memory management aspects of operating systems)"
    }
    ]
  }
```

Listing 2: The content of 84.json

I generated the ranking list using Galago from the command line:

```
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago batch-
    search --index=./index --requested=10 84.json
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2897.html 1 -6.20214617 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2033.html 2 -6.28636814 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1947.html 3 -6.36386467 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2559.html 4 -6.45008887 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2491.html 5 -6.45483869 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2680.html 6 -6.52951010 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2701.html 7 -6.54338217 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1886.html 8 -6.54934530 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2944.html 9 -6.56475305 galago
CACM-13 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2537.html 10 -6.58501681 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2297.html 1 -5.78832924 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1752.html 2 -5.98368977 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2902.html 3 -6.02077716 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2319.html 4 -6.04039597 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2406.html 5 -6.04959217 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2740.html 6 -6.07650436 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-1725.html 7 -6.08352717 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2358.html 8 -6.08425902 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2988.html 9 -6.10546471 galago
CACM-27 Q0 /home/hussam/Desktop/galago/galago-bin/bin/cacm/CACM-2357.html 10 -6.13047371 galago
```

I downloaded the relevance judgment file "cacm.rel". The section for query 13 and 27 is:

```
13 Q0 CACM-115 1
13 Q0 CACM-1223 1
13 Q0 CACM-1231 1
13 Q0 CACM-1551 1
13 Q0 CACM-1625 1
13 Q0 CACM-1795 1
```

```
13 Q0 CACM-1807 1
13 Q0 CACM-1947 1
13 Q0 CACM-2495 1
13 Q0 CACM-2579 1
13 Q0 CACM-2897 1
27 Q0 CACM-1641 1
27 Q0 CACM-1642 1
27 Q0 CACM-1750 1
27 Q0 CACM-1752 1
27 Q0 CACM-1879 1
27 Q0 CACM-1884 1
27 Q0 CACM-1901 1
27 Q0 CACM-2095 1
27 Q0 CACM-2297 1
27 Q0 CACM-2435 1
27 Q0 CACM-2481 1
27 Q0 CACM-2498 1
27 Q0 CACM-2560 1
27 Q0 CACM-2596 1
27 Q0 CACM-2669 1
27 Q0 CACM-2734 1
27 Q0 CACM-2747 1
27 Q0 CACM-2768 1
27 Q0 CACM-2798 1
27 Q0 CACM-2818 1
27 Q0 CACM-2859 1
27 Q0 CACM-2864 1
27 Q0 CACM-2902 1
27 Q0 CACM-2918 1
27 Q0 CACM-2955 1
27 Q0 CACM-2983 1
27 Q0 CACM-2988 1
27 Q0 CACM-3000 1
27 Q0 CACM-3052 1
```

Therefore:
The number of relevant documents in the collection for query #13 is 11.
The number of relevant documents in the collection for query #27 is 29.
Each document in the result set can be identified as a "Hit" or "Miss".
Precision = Number of relevant documents retrieved / number of retrieved documents
For query #13:

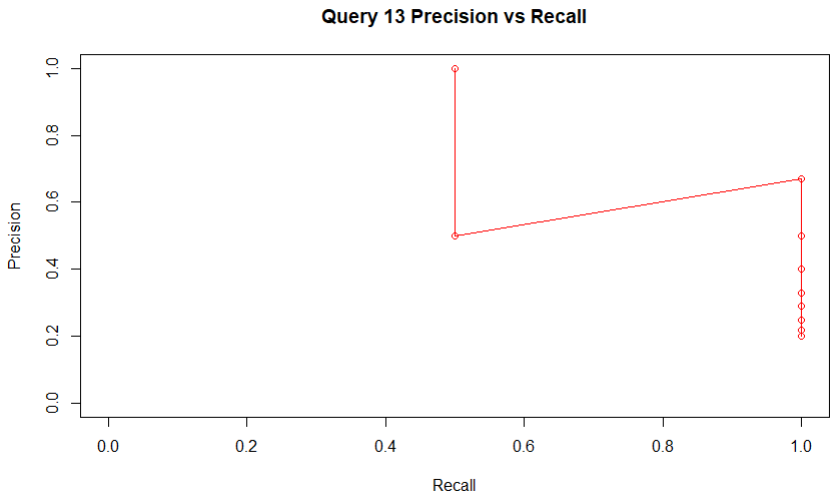
Result	File Name	Hit/Miss	Precision	Recall
1	CACM-2897.html	Hit	1.0	0.5
2	CACM-2033.html	Miss	0.5	0.5
3	CACM-1947.html	Hit	0.67	1.0
4	CACM-2559.html	Miss	0.5	1.0
5	CACM-2491.html	Miss	0.4	1.0
6	CACM-2680.html	Miss	0.33	1.0
7	CACM-2701.html	Miss	0.29	1.0
8	CACM-1886.html	Miss	0.25	1.0
9	CACM-2944.html	Miss	0.22	1.0
10	CACM-2537.html	Miss	0.2	1.0

It is time to plot uninterpolated Precision vs Recall for query #13.
The points are saved in the file "q13.csv". I used R to create the graph.

Uninterpolated:

```
q13 <- read.csv("q13.csv", header = TRUE)
plot(q13$Recall, q13$Precision, type = "o", col="red" , xlab = "Recall", ylab = "Precision", main = "Query
    13 Precision vs Recall", xlim = c(0,1), ylim = c(0,1))
```

Figure 1: Uninterpolated Precision VS Recall for Query 13



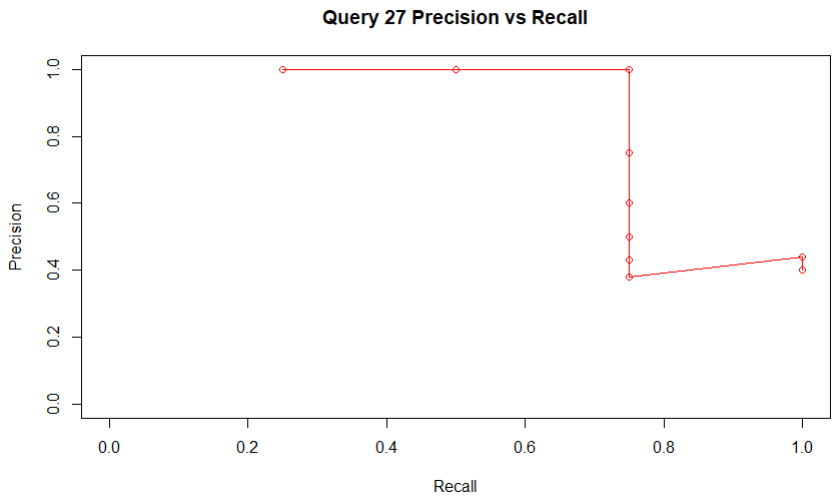
For query #27:

Result	File Name	Hit/Miss	Precision	Recall
1	CACM-2297.html	Hit	1.0	0.25
2	CACM-1752.html	Hit	1.0	0.0.5
3	CACM-2902.html	Hit	1.0	0.75
4	CACM-2319.html	Miss	0.75	0.75
5	CACM-2406.html	Miss	0.6	0.75
6	CACM-2740.html	Miss	0.5	0.75
7	CACM-1725.html	Miss	0.43	0.75
8	CACM-2358.html	Miss	0.38	0.75
9	CACM-2988.html	Hit	0.44	1.0
10	CACM-2357.html	Miss	0.4	1.0

It is time to plot uninterpolated Precision vs Recall for query #27.
The points are saved in the file "q27.csv". I used R to create the graph.
Uninterpolated:

```
q27 <- read.csv("q27.csv", header = TRUE)
plot(q27$Recall, q27$Precision, type = "o", col="red" , xlab = "Recall", ylab = "Precision", main = "Query
    27 Precision vs Recall", xlim = c(0,1), ylim = c(0,1))
```

Figure 2: Uninterpolated Precision VS Recall for Query 27



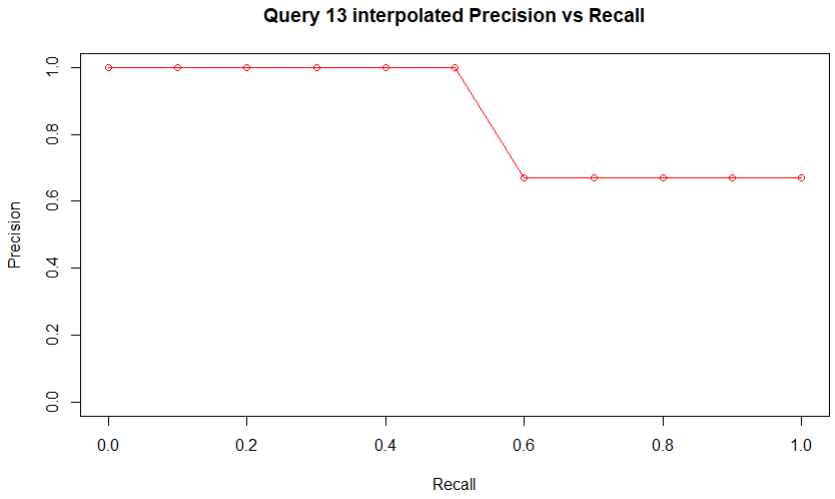
Interpolated: It's time to create a table of interpolated precision values at standard recall levels. By definition, the interpolated precision at any recall level is the maximum precision observed in any recall-precision point at a higher recall level.

Recall	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Q13	1.0	1.0	1.0	1.0	1.0	1.0	0.67	0.67	0.67	0.67	0.67
Q27	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.75	0.44	0.44
AVG	1.0	1.0	1.0	1.0	1.0	1.0	0.84	0.84	0.71	0.56	0.56

```
q13 <- read.csv("iq13.csv", header = TRUE)
```

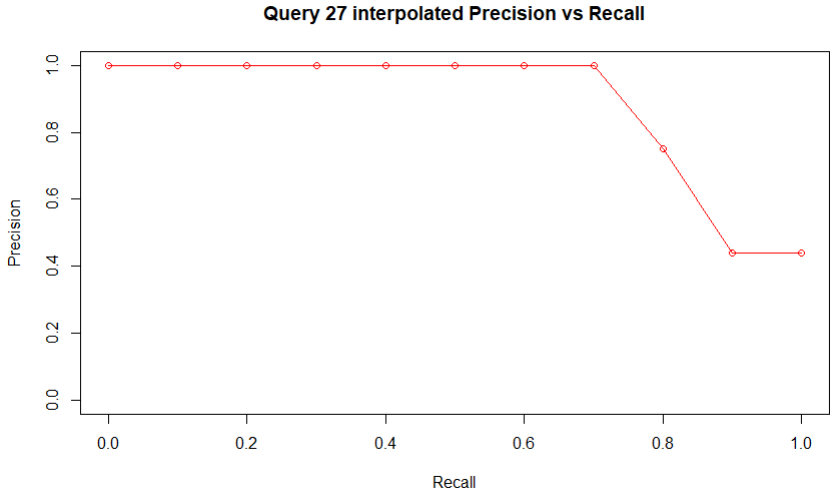
```
plot(q13$Recall, q13$Precision, type = "o", col="red" , xlab = "Recall", ylab = "Precision", main = "Query 13 interpolated Precision vs Recall", xlim = c(0,1), ylim = c(0,1))
```

Figure 3: Interpolated Precision VS Recall for Query 13




```
q27 <- read.csv("iq27.csv", header = TRUE)
plot(q27$Recall, q27$Precision, type = "o", col="red" , xlab = "Recall", ylab = "Precision", main = "Query 27 interpolated Precision vs Recall", xlim = c(0,1), ylim = c(0,1))
```

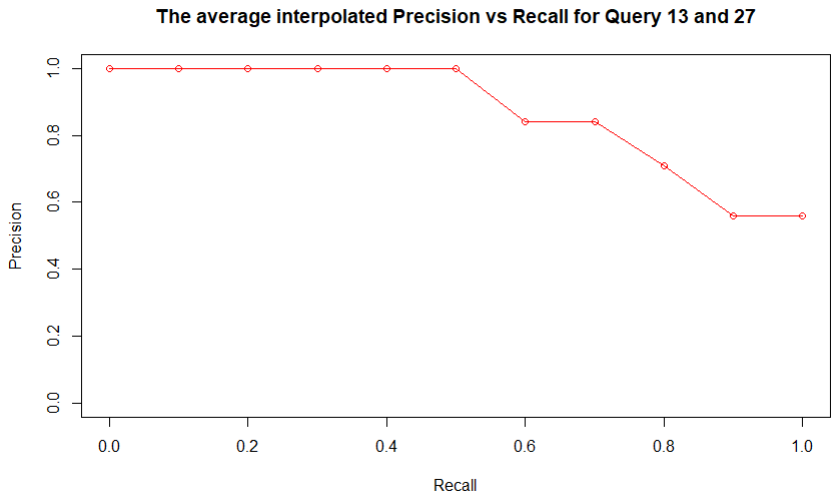
Figure 4: Interpolated Precision VS Recall for Query 27



The average interpolated recall-precision graph:

```
a <- read.csv("a.csv", header = TRUE)
plot(a$Recall, a$Precision, type = "o", col="red" , xlab = "Recall", ylab = "Precision", main = "The average interpolated Precision vs Recall for Query 13 and 27", xlim = c(0,1), ylim = c(0,1))
```

Figure 5: The average interpolated Precision vs Recall for Query 13 and 27



Question 3:

Exercise 8.5: Generate the mean average precision, recall-precision graph, average NDCG at 5 and 10, and precision at 10 for the entire CACM query set.

Answer:

Galago can generate different measures for an entire set using the "eval" command. I am using the newer version of Galago, which only supports JSON files as input file of queries. Queries are available in two different formats, raw and processed. They are saved in "cacm.query" and "cacm.query.xml" on this page:

http://www.search-engines-book.com/collections/

I downloaded both files and noticed that they do not have the same queries. Query #64 is missing from the XML file. I went and added the last query to the XML file and then I used an online XML to JSON converter to convert the XML file to JSON. The file "85.json" contains all 64 queries in JSON format and it is ready to use with Galago batch-search command. I ran the batch-search and saved the output to the file "85.out".

```
1 hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago batch-
      search --index=index --requested=10 85.json > 85.out
```

The batch search was processed successfully and the output was saved to "85.out".

I tried to calculate the requested measures using the "eval command" of the baseline file "85.out" against the relevance judgment file "cacm.rel", but I ran into a problem. The command "eval" is being executed properly and the values for the number of retrieved documents and the relevant documents is correct, but the number of retrieved relevant document is coming up as zero for all queries. This is making all values for different measures a zero, which I know is not correct from questions 1 and 2 which I did by hand.

```
1  hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago eval --
       judgments=cacm.rel --baseline=85.out
2  num_ret                    all   520.00000
3  num_rel                    all   796.00000
4  num_rel_ret                all  0.00000
5  num_unjug_ret@20           all   520.00000
6  map                        all  0.00000
7  R-prec                     all  0.00000
8  bpref                      all  0.00000
9  recip_rank                 all  0.00000
10 ndcg                       all  0.00000
11 ndcg5                      all  0.00000
12 ndcg10                     all  0.00000
13 ndcg20                     all  0.00000
14 ERR                        all  0.00000
15 ERR10                      all  0.00000
16 ERR20                      all  0.00000
17 P5                         all  0.00000
18 P10                        all  0.00000
19 P15                        all  0.00000
20 P20                        all  0.00000
21 P30                        all  0.00000
22 P100                       all  0.00000
23 P200                       all  0.00000
24 P500                       all  0.00000
25 P1000                      all  0.00000
```

I was unable to generate the requested graphs because I am getting incorrect results. I realized that I need this to work to solve parts of other questions in the assignment as well. Generating the graphs is rather simple using R just like I did in question 2.

Question 4:

Exercise 8.7: Another measure that has been used in number of evaluations is R-precision. This is defined as the precision at R documents, where R is the number of relevant documents for a query. It is used in situations where there is a large variation in the number of relevant documents per query. Calculate the average R-precision for the CACM query set and compare it to the other measures.

Answer:

I found the number of relevant documents for each query from the relevance judgment file using the "grep" command.

```
1 hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ cat cacm.rel |
      grep "^1 Q0" | wc -l
2 5
```

I repeated the command for all 64 queries. The following table contains the number of relevant documents for each query.

Query Number	Number of relevant documents
1	5
2	3
3	6
4	12
5	8
6	3
7	28
8	3
9	9
10	35
11	19
12	5
13	11
14	44
15	10
16	17
17	16
18	11
19	11
20	3
21	11
22	17
23	4
24	13
25	51
26	30
27	29
28	5
29	19

30	4
31	2
32	3
33	1
34	0
35	0
36	20
37	12
38	16
39	12
40	10
41	0
42	21
43	41
44	17
45	26
46	0
47	0
48	12
49	8
50	0
51	0
52	0
53	0
54	0
55	0
56	0
57	1
58	30
59	43
60	27
61	31
62	8
63	12
64	1

There is a large variation in the number of relevant documents per query. The minimum is 0 document, and the maximum is 51 document.

I used the "batch-search" command in Galago to fetch 51 results for each query.

```
1 hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago batch-
      search --index=index --requested=51 cacm.query.json > 87.out
```

The batch search was processed successfully and the result can be seen in the file named "87.out", however, I ran into a problem when I tried to evaluate it against the relevance judgment file "cacm.rel". The command "eval" is being executed properly and the values for the number of retrieved documents and the relevant documents is correct, but the number of retrieved relevant document is coming up as zero for all queries. This is making all values for different measures a zero. The R-Precision value, as well as other measures, according

to Galago evaluation is zero for all queries which I know is not correct from questions 1 and 2 which I did by hand.

```
1  hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/galago/galago-bin/bin$ ./galago eval --
       judgments=cacm.rel --baseline=87.out --details=true
2  num_ret                  64   51.00000
3  num_rel                  64   1.00000
4  num_rel_ret              64   0.00000
5  num_unjug_ret@20         64   20.00000
6  map                      64   0.00000
7  R-prec                   64   0.00000
8  bpref                    64   0.00000
9  recip_rank               64   0.00000
10 ndcg                     64   0.00000
11 ndcg5                    64   0.00000
12 ndcg10                   64   0.00000
13 ndcg20                   64   0.00000
14 ERR                      64   0.00000
15 ERR10                    64   0.00000
16 ERR20                    64   0.00000
17 P5                       64   0.00000
18 P10                      64   0.00000
19 P15                      64   0.00000
20 P20                      64   0.00000
21 P30                      64   0.00000
22 P100                     64   0.00000
23 P200                     64   0.00000
24 P500                     64   0.00000
25 P1000                    64   0.00000
26 num_ret                  all  2652.00000
27 num_rel                  all  796.00000
28 num_rel_ret              all  0.00000
29 num_unjug_ret@20         all  1040.00000
30 map                      all  0.00000
31 R-prec                   all  0.00000
32 bpref                    all  0.00000
33 recip_rank               all  0.00000
34 ndcg                     all  0.00000
35 ndcg5                    all  0.00000
36 ndcg10                   all  0.00000
37 ndcg20                   all  0.00000
38 ERR                      all  0.00000
39 ERR10                    all  0.00000
40 ERR20                    all  0.00000
41 P5                       all  0.00000
42 P10                      all  0.00000
43 P15                      all  0.00000
44 P20                      all  0.00000
45 P30                      all  0.00000
46 P100                     all  0.00000
47 P200                     all  0.00000
48 P500                     all  0.00000
49 P1000                    all  0.00000
```

Question 5:

Exercise 8.9: For one query in the CACM collection, generate a ranking and calculate BPREF. Show that the two formulations of BPREF give the same value.

Answer:

I used query 13 that I used for question 2. The steps to find the first 10 results for the query and the relevant/non-relevant documents are identical to the steps in question 2.

Results for query #13:

Result	File Name	Hit/Miss
1	CACM-2897.html	Hit
2	CACM-2033.html	Miss
3	CACM-1947.html	Hit
4	CACM-2559.html	Miss
5	CACM-2491.html	Miss
6	CACM-2680.html	Miss
7	CACM-2701.html	Miss
8	CACM-1886.html	Miss
9	CACM-2944.html	Miss
10	CACM-2537.html	Miss

It's time to calculate BPREF by hand using the two definitions of BPREF:

$$BPREF = \frac{1}{R}\sum_{d_r}(1 - \frac{N_{dr}}{R})$$

And

$$BPREF = \frac{P}{P+Q}$$

Where R is the number of relevant documents, d_r is a relevant document, and N_{dr} gives the number of non-relevant documents (from the set of R non-relevant documents that are considered) that are ranked higher than d_r

R = 2, therefore the number of non-relevant documents that are considered is 2.

$$BPREF = \frac{1}{2}[(1 - 0/2) + (1 - 1/2)] = \frac{3}{4} = 0.75$$

Using the alternative definition:

$$BPREF = \frac{2}{2+1} = \frac{2}{3} = 0.67$$

I have different values at the end. I tried different definitions that I found online, but came up with results that are much different than this.

Question 6:

Exercise 9.8: Cluster the following set of two-dimensional instances into three clusters using each of the five agglomerative clustering methods:

$$(-4,-2),(-3,-2),(-2,-2),(-1,-2),(1,-1),(1,1),(2,3),(3,2),(3,4),(4,3)$$

Discuss the differences in the clusters across methods. Which methods produce the same clusters? How do these clusters compare to how you would manually cluster the points?

Answer:

I wrote a simple python script to cluster the data points using each of the five agglomerative clustering methods (single, complete, average, centroid, and ward). The script "cluster5.py" clusters the points into 3 clusters. It prints out each point and its cluster number.

```
import numpy as np
import scipy.cluster.hierarchy as hac

a = np.array([[-4, -2],
              [-3, -2],
              [-2, -2],
              [-1, -2],
              [1, -1],
              [1, 1],
              [2, 3],
              [3, 2],
              [3, 4],
              [4, 3]])

for method in ['single', 'complete', 'average', 'centroid', 'ward', 'weighted', 'median']:
    z = hac.linkage(a, method=method)
    z1 = hac.fcluster(z, 3, criterion='maxclust')
    print 'Clustering Method: ', method
    for i in range(0, len(z1)):
        print 'Cluster: ', z1[i], a[i]
```

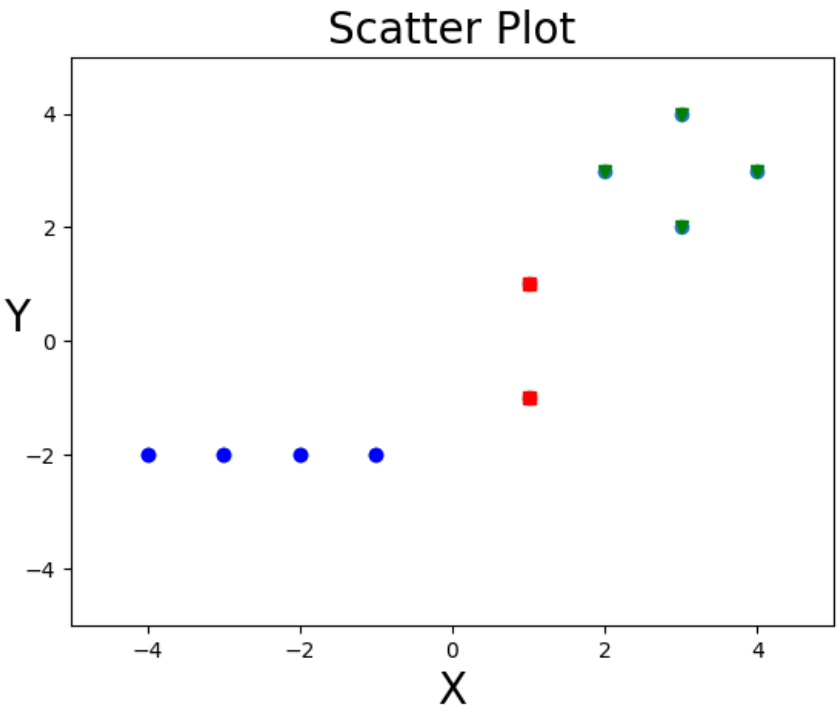
Listing 3: The content of cluster5.py

```
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/A4/Q6$ python cluster5.py
Clustering Method: single
Cluster:  2 [-4 -2]
Cluster:  2 [-3 -2]
Cluster:  2 [-2 -2]
Cluster:  2 [-1 -2]
Cluster:  3 [ 1 -1]
Cluster:  3 [1 1]
Cluster:  1 [2 3]
Cluster:  1 [3 2]
Cluster:  1 [3 4]
Cluster:  1 [4 3]
Clustering Method: complete
Cluster:  1 [-4 -2]
Cluster:  1 [-3 -2]
Cluster:  1 [-2 -2]
Cluster:  1 [-1 -2]
Cluster:  2 [ 1 -1]
Cluster:  2 [1 1]
Cluster:  3 [2 3]
Cluster:  3 [3 2]
Cluster:  3 [3 4]
Cluster:  3 [4 3]
Clustering Method: average
Cluster:  1 [-4 -2]
```

```
Cluster:  1 [-3 -2]
Cluster:  1 [-2 -2]
Cluster:  1 [-1 -2]
Cluster:  3 [ 1 -1]
Cluster:  3 [1  1]
Cluster:  2 [2  3]
Cluster:  2 [3  2]
Cluster:  2 [3  4]
Cluster:  2 [4  3]
Clustering Method: centroid
Cluster:  1 [-4 -2]
Cluster:  1 [-3 -2]
Cluster:  1 [-2 -2]
Cluster:  1 [-1 -2]
Cluster:  3 [ 1 -1]
Cluster:  3 [1  1]
Cluster:  2 [2  3]
Cluster:  2 [3  2]
Cluster:  2 [3  4]
Cluster:  2 [4  3]
Clustering Method: ward
Cluster:  1 [-4 -2]
Cluster:  1 [-3 -2]
Cluster:  1 [-2 -2]
Cluster:  1 [-1 -2]
Cluster:  3 [ 1 -1]
Cluster:  3 [1  1]
Cluster:  2 [2  3]
Cluster:  2 [3  2]
Cluster:  2 [3  4]
Cluster:  2 [4  3]
Clustering Method: weighted
Cluster:  1 [-4 -2]
Cluster:  1 [-3 -2]
Cluster:  1 [-2 -2]
Cluster:  1 [-1 -2]
Cluster:  3 [ 1 -1]
Cluster:  3 [1  1]
Cluster:  2 [2  3]
Cluster:  2 [3  2]
Cluster:  2 [3  4]
Cluster:  2 [4  3]
Clustering Method: median
Cluster:  1 [-4 -2]
Cluster:  1 [-3 -2]
Cluster:  1 [-2 -2]
Cluster:  1 [-1 -2]
Cluster:  3 [ 1 -1]
Cluster:  3 [1  1]
Cluster:  2 [2  3]
Cluster:  2 [3  2]
Cluster:  2 [3  4]
Cluster:  2 [4  3]
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/A4/Q6$
```

I used a python library, "scipy" to cluster the points. After getting the same results for all 5 clustering methods, I tried two other methods, weighted and median, to see if I get different results, but I did not. Since I am using scipy library as a black box, I was not sure if I am supposed to get the same results. This is how I would manually cluster the points, which is the same result I have from all 7 clustering methods.

Figure 6: Clustering points into 3 clusters



Question 7:

Exercise 9.9: Use K-means and spherical K-means to cluster the data points in Exercise 9.8. How do the clusterings differ?

Answer:

The data points from Exercise 9.8 are:

$$(-4,-2),(-3,-2),(-2,-2),(-1,-2),(1,-1),(1,1),(2,3),(3,2),(3,4),(4,3)$$

I wrote two simple python scripts to cluster the data points, "k.py" and "sk.py", using K-means and spherical K-means respectively. I ran both programs for k = 2, 3, and 4.

The programs also generate plots for the data before and after clustering.

0.1 K-means:

```
import sys
from sklearn.cluster import KMeans
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt

if len(sys.argv) != 2:
    print "Usage: python k.py <K value>"
    print "Example: python k.py 3"
    exit()


x = np.array([-4, -3, -2, -1, 1, 1, 2, 3, 3, 4])
y = np.array([-2, -2, -2, -2, -1, 1, 3, 2, 4, 3])

plt.plot()
plt.xlim([-5, 5])
plt.ylim([-5, 5])
plt.title('Scatter Plot', fontsize=20)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', rotation=0, fontsize=20)
plt.scatter(x, y)
plt.show()

# create new plot and data
plt.plot()
X = np.array(list(zip(x, y))).reshape(len(x), 2)
colors = ['b', 'g', 'r', 'c']
markers = ['o', 'v', 's', 'h']

# KMeans algorithm
K = int(sys.argv[1])
kmeans_model = KMeans(n_clusters=K).fit(X)

plt.plot()
for i, l in enumerate(kmeans_model.labels_):
    plt.plot(x[i], y[i], color=colors[l], marker=markers[l],ls='None')
    plt.xlim([-5, 5])
    plt.ylim([-5, 5])

plt.title('Scatter Plot', fontsize=20)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', rotation=0, fontsize=20)
plt.scatter(x, y)
plt.show()
```

Listing 4: The content of k.py

```
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/99$ python k.py 2
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/99$ python k.py 3
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/99$ python k.py 4
```

Figure 7: K-means for K = 1

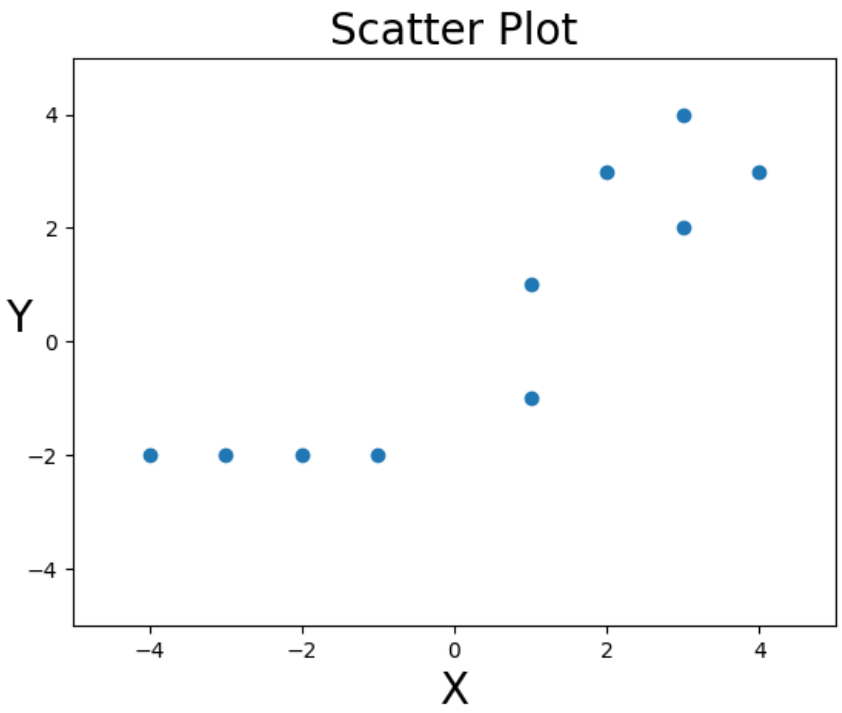


Figure 8: K-means for K = 2

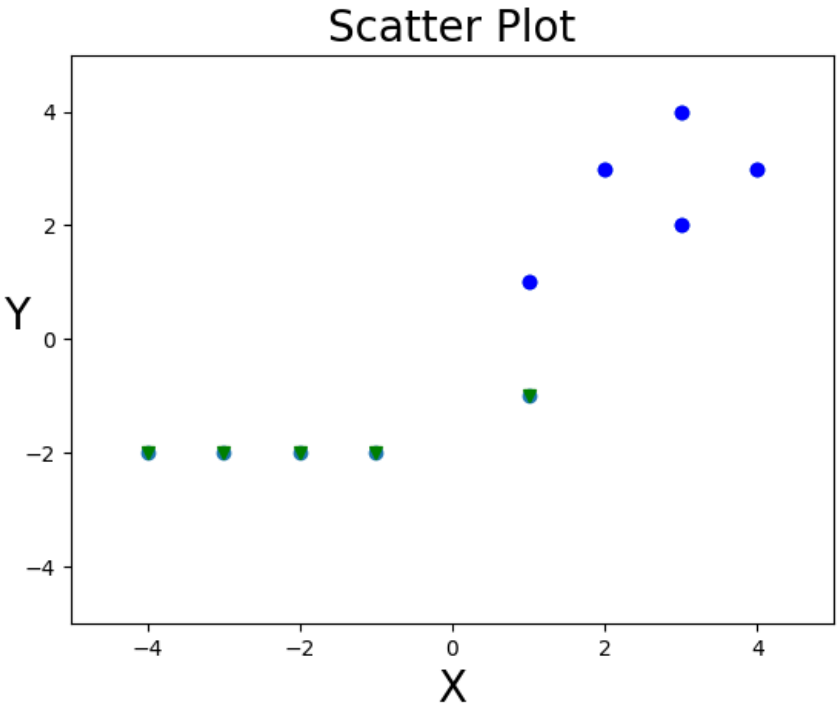


Figure 9: K-means for K = 3

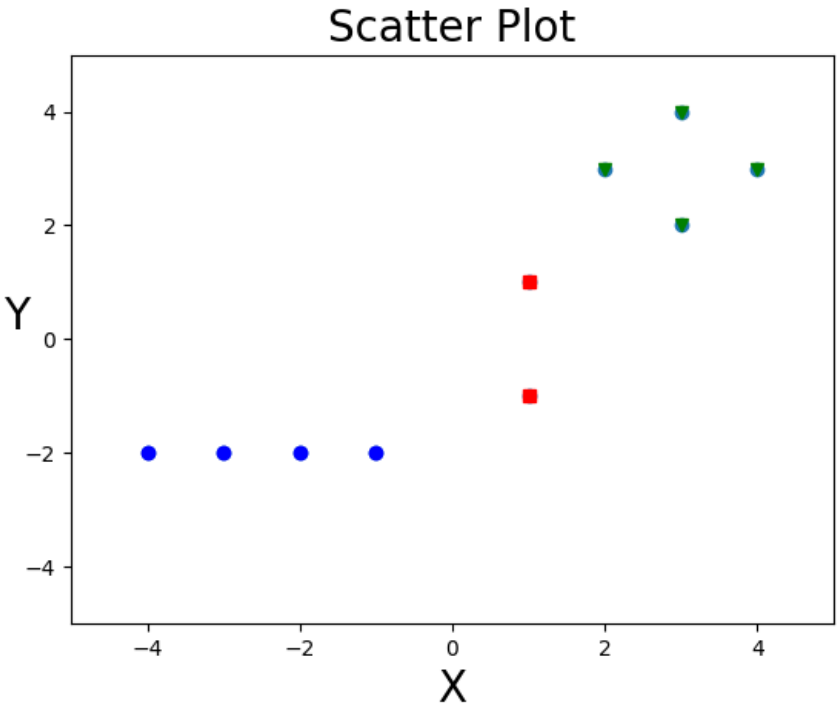
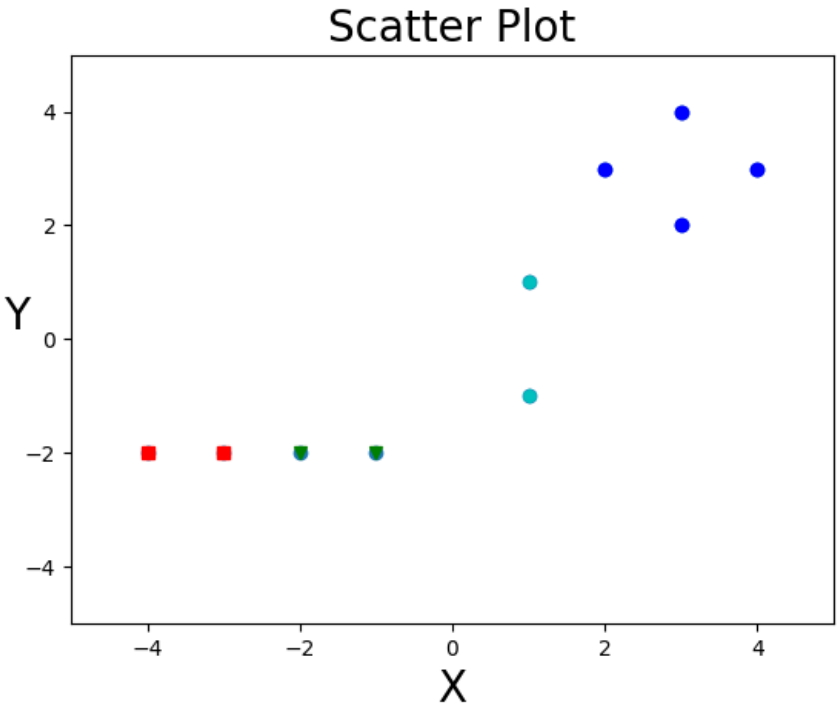


Figure 10: K-means for K = 4



0.2 Spherical K-means:

```
import sys
from sklearn.cluster import KMeans
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt

if len(sys.argv) != 2:
    print "Usage: python sk.py <K value>"
    print "Example: python sk.py 3"
    exit()


x = np.array([-4, -3, -2, -1, 1, 1, 2, 3, 3, 4])
y = np.array([-2, -2, -2, -2, -1, 1, 3, 2, 4, 3])

plt.plot()
plt.xlim([-5, 5])
plt.ylim([-5, 5])
plt.title('Scatter Plot', fontsize=20)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', rotation=0, fontsize=20)
plt.scatter(x, y)
plt.show()

# create new plot and data
plt.plot()
X = np.array(list(zip(x, y))).reshape(len(x), 2)
colors = ['b', 'g', 'r', 'c']
markers = ['o', 'v', 's', 'h']

# KMeans algorithm
K = int(sys.argv[1])
from spherecluster import SphericalKMeans
kmeans_model = SphericalKMeans(n_clusters=K).fit(X)

plt.plot()
for i, l in enumerate(kmeans_model.labels_):
    plt.plot(x[i], y[i], color=colors[l], marker=markers[l],ls='None')
    plt.xlim([-5, 5])
    plt.ylim([-5, 5])

plt.title('Scatter Plot', fontsize=20)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', rotation=0, fontsize=20)
plt.scatter(x, y)
plt.show()
```

Listing 5: The content of sk.py

```
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/99$ python sk.py 4
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/99$ python sk.py 3
hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/99$ python sk.py 2
```

Figure 11: Spherical K-means for K = 2

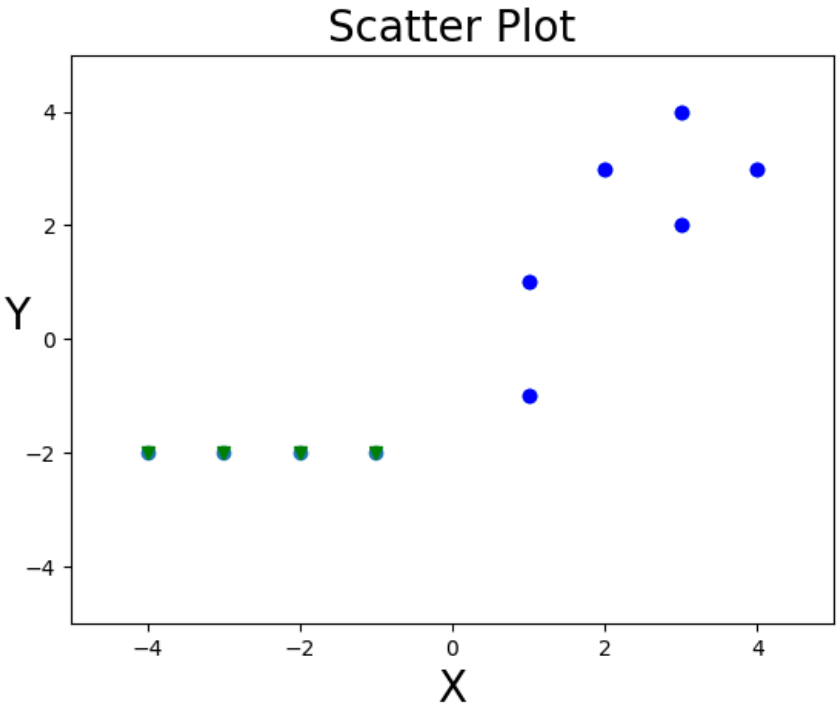


Figure 12: Spherical K-means for K = 3

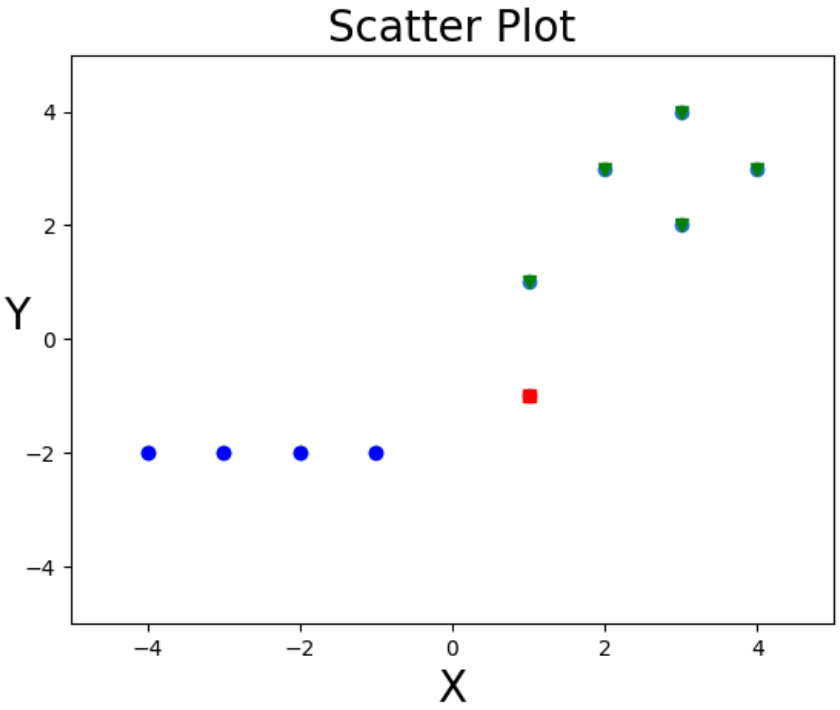
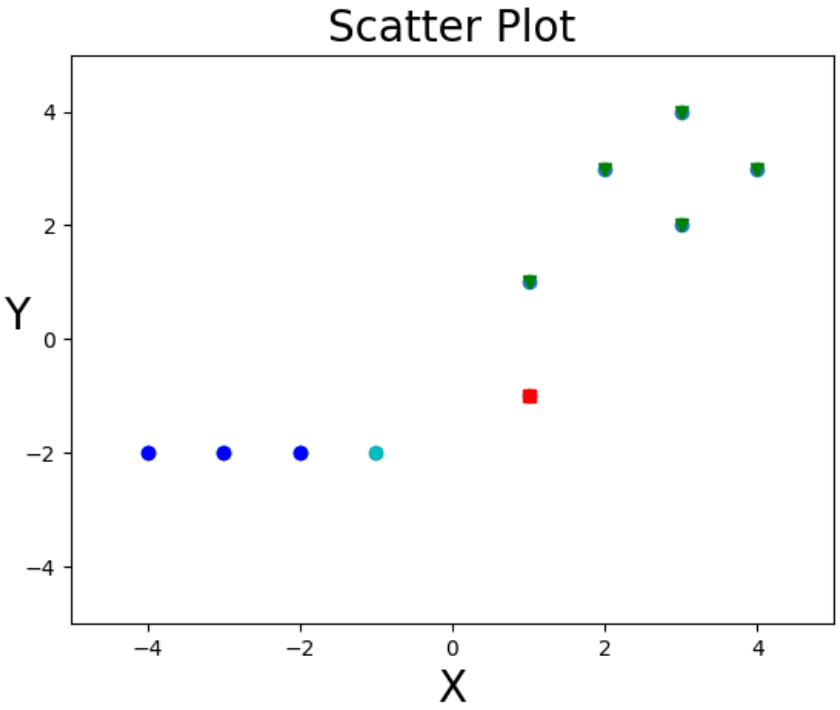


Figure 13: Spherical K-means for K = 4



Observation:

K-means and Spherical K-means produced different results for all tested K values. The reason is that each clustering algorithm has a different approach to solve the clustering problem. K-means minimizes the Euclidean distance between the cluster center and the members of the cluster. The intuition behind this is that the radial distance from the cluster center to the element location should be similar for all elements of that cluster. On the other hand, Sperical K-means sets the center of the cluster such that it makes both uniform and minimal angle between components. The points should have consistent spacing between each other. That spacing is simpler to quantify as cosine similarity.

Question 8:

Exercise 9.11: The K nearest neighbors of a document could be represented by links to those documents. Describe two ways this representation could be used in a search application.

Answer:

One way this representation could be used in a search application is by having a section for documents that are related to the results in which the neighbors of a result could be listed. This could be beneficial if the assumption that related pages link to each other is true.

A second way this representation could be useful is for narrowing down the search results. The user could choose to search in a particular cluster after getting results from different clusters.

References

[1] Stackoverflow. https://stackoverflow.com/questions/tagged/python.

[2] http://convertjson.com/xml-to-json.htm

[3] https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html

[4] https://sourceforge.net/p/lemur/wiki/Galago%20Query%20Language/?version=2

[5] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot