# Assignment 4

CS834-F16: Introduction to Information Retrieval

Fall 2016

Erika Siregar

CS Department - Old Dominion University

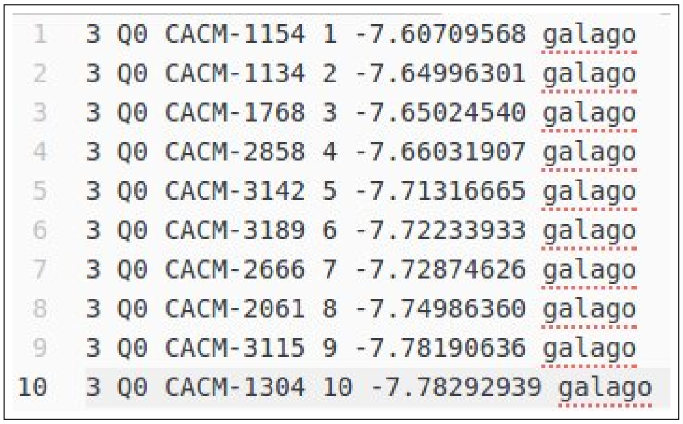
December 8, 2016

# Question 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

For this assignment, I choose the query no. 3: 'intermediate languages used in construction of multi targeted compilers tcoll'. This query is taken from CACM collection that is available at `http://www.search-engines-book.com/collections/`. Based on this query, I generate a ranking using Galago 3.10 [1]. I took the top 10 rank positions and follow the guide from the textbook [2]. Figure 1 shows the result of the ranking generated using galago.



```
3 Q0 CACM-1154 1 -7.60709568 galago
3 Q0 CACM-1134 2 -7.64996301 galago
3 Q0 CACM-1768 3 -7.65024540 galago
3 Q0 CACM-2858 4 -7.66031907 galago
3 Q0 CACM-3142 5 -7.71316665 galago
3 Q0 CACM-3189 6 -7.72233933 galago
3 Q0 CACM-2666 7 -7.72874626 galago
3 Q0 CACM-2061 8 -7.74986360 galago
3 Q0 CACM-3115 9 -7.78190636 galago
3 Q0 CACM-1304 10 -7.78292939 galago
```

Figure 1: Top 10 Rank Positions for Query no. 3 generated by Galago

After obtaining this ranking, we are now ready to compute the metrics that are required. In this question, there are 5 metrics that will be calculated:

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

### Precision, Recall, and Average Precision

To get the average precision value, we first need to compute recall and precision for each document that is retrieved. Recall is the proportion of relevant documents that are retrieved, and precision is the proportion of retrieved documents that are relevant [2]. Recall and precision are calculated using these following formulas:

$$Recall = \frac{|A \cap B|}{|A|} \tag{1}$$

$$Precision = \frac{|A \cap B|}{|B|} \tag{2}$$

To get the average precision value, we only need to calculate the average of the precision values from all relevant documents.

### NDCG

Do the following steps to calculate NDCG:

1. Set a relevance level for each document. For this assignment, I use boolean code: 1 if the document is relevant and 0 otherwise.

2. Calculate the DCG using this formula:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i} \tag{3}$$

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

3. Create the ideal ranks. In my opinion, it is as if we sort the relevance level in the non-increasing order.

4. Calculate IDCG in the same way as we do calculation for DCG in point no. 2.

5. Finally, calculate NDCG for p = 5 and p = 10 using this formula:

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{4}$$

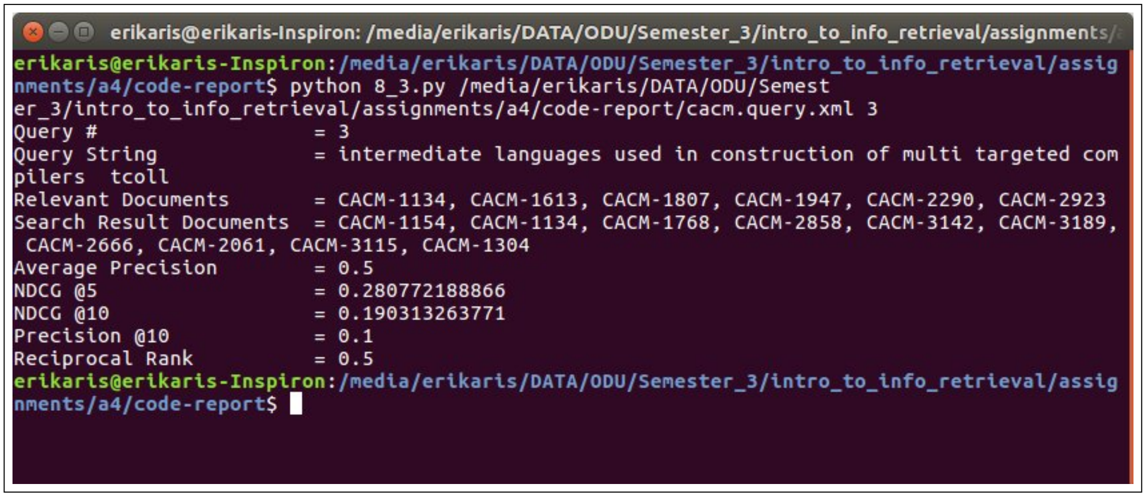
### Precision at 10

This is similar to the precision that we have discussed in the previous section, but we only consider the precision at rank = 10.

### Reciprocal Rank

Reciprocal rank can be defined as 1 divided by 'the rank at which the first relevant document is found'. For example, if the first relevant document is found at rank = 2, then the reciprocal rank is $\frac{1}{2}$.

Figure 2 shows the value of average precision, NDCG at 5, NDCG at 10, precision at 10, and the reciprocal rank for query 3. The code to generate this metrics is also can be seen at listing 1. I also create a library named 'galago.py' which is a 'wrapper' of Galago tools and also contains the calculation for the metrics that are needed in this assignment. The code for galago.py can be seen on listing 2



```
erikaris@erikaris-Inspiron: /media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assig
nments/a4/code-report$ python 8_3.py /media/erikaris/DATA/ODU/Semest
er_3/intro_to_info_retrieval/assignments/a4/code-report/cacm.query.xml 3
Query #                  = 3
Query String             = intermediate languages used in construction of multi targeted com
pilers  tcoll
Relevant Documents       = CACM-1134, CACM-1613, CACM-1807, CACM-1947, CACM-2290, CACM-2923
Search Result Documents  = CACM-1154, CACM-1134, CACM-1768, CACM-2858, CACM-3142, CACM-3189,
 CACM-2666, CACM-2061, CACM-3115, CACM-1304
Average Precision        = 0.5
NDCG @5                  = 0.280772188866
NDCG @10                 = 0.190313263771
Precision @10            = 0.1
Reciprocal Rank          = 0.5
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assig
nments/a4/code-report$ █
```

Figure 2: The summary of metrics for query 3

```python
#!/usr/bin/python

import errno
import os
from optparse import OptionParser
from lib.galago import GalagoRank

if __name__ == '__main__':
parser = OptionParser(description='Generate a ranking using Galago')
parser.set_usage(parser.get_usage().replace('\n', '') + ' <xml_file_input> [q1 ... qn]')
parser.add_option('-g', '--galago', dest="galago_bin", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/galago/galago-3.10-bin/bin/galago',
help='Galago "home" directory')
parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm',
help='Document directory to be indexed')
parser.add_option('-j', '--judgements', dest="judgements_file", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm.rel',
help='File .rel as galago eval judgments')
parser.add_option('-r', '--result', dest="result_count", default='10', help='Number of result')

(options, args) = parser.parse_args()
options = vars(options)

if len(args) < 2:
parser.print_help()
exit()


out_dir = os.path.abspath('output')

try: os.makedirs(out_dir)
except OSError, e:
if e.errno != errno.EEXIST: raise

index_dir = os.path.join(os.path.pardir, 'index')
```

```
xml_query_file = os.path.abspath(args[0])
json_query_file = os.path.join(out_dir, 'query.json')
rel_file = os.path.join(out_dir, 'result.rel')
res_file = os.path.join(out_dir, 'result.res')
eval_file = os.path.join(out_dir, 'result.eval')

q_ids = []
if len(args) > 1:
q_ids = args[1:]

galago = GalagoRank(options['galago_bin'], options['judgements_file'])
if galago.index(options['document_dir'], index_dir):
for q_id in q_ids:
json_query = galago.build_json_input(xml_query_file, q_id, json_query_file)

rel_docs = galago.get_relevance_docs(q_id)
res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count'
    ])

galago.eval(options['judgements_file'], res_file, eval_file)

print 'Query #                  = {}'.format(json_query['number'])
print 'Query String             = {}'.format(json_query['text'])
print 'Relevant Documents       = {}'.format(', '.join(rel_docs))
print 'Search Result Documents  = {}'.format(', '.join(res_docs))
print 'Average Precision        = {}'.format(galago.get_map(rel_docs, res_docs))
print 'NDCG @5                  = {}'.format(galago.get_ndcg(5, rel_docs, res_docs))
print 'NDCG @10                 = {}'.format(galago.get_ndcg(10, rel_docs, res_docs))
print 'Precision @10            = {}'.format(galago.get_all_precisions(rel_docs,
    res_docs)[9])
print 'Reciprocal Rank          = {}'.format(galago.get_reciprocal_rank(rel_docs,
    res_docs))
```

Listing 1: Code for question 8.3

```
#!/usr/bin/python

import json
import os
from math import log
from subprocess import Popen, PIPE, call
from threading import Thread
import xmltodict


class Command(object):
def __init__(self, cmd, out_pipe_callback=None, err_pipe_callback=None):
self.cmd = cmd
self.process = None
self.out_pipe_callback = out_pipe_callback
self.err_pipe_callback = err_pipe_callback

def run(self, timeout, args=()):
def target():
self.process = Popen(self.cmd, stdout=PIPE, stderr=PIPE)

if self.out_pipe_callback:
stdout_thread = Thread(target=self.out_pipe_callback,
args=(self.process.stdout, ) + args)
stdout_thread.daemon = True
stdout_thread.start()

if self.err_pipe_callback:
stderr_thread = Thread(target=self.err_pipe_callback,
args=(self.process.stderr, ) + args)
stderr_thread.daemon = True
stderr_thread.start()

self.process.wait()

thread = Thread(target=target)
thread.daemon = True
thread.start()

thread.join(timeout)
try: self.process.terminate()
except: pass

return self.process.returncode


class GalagoRank(object):
galago_bin = None
rel_docs = {}
res_docs = []

def __init__(self, galago_bin, judgements_file):
self.galago_bin = galago_bin
self.rel_docs = self.build_relevance(judgements_file)

def index(self, document_dir, index_dir):
if not os.path.exists(index_dir):
bash = '"{}" build --indexPath="{}" --inputPath="{}"'.format(
self.galago_bin, index_dir, document_dir)

code = call(['bash', '-c', bash])
return code == 0

else:
return True

def build_relevance(self, judgements_file):
with open(judgements_file, 'r') as fp:
rel_docs = {}
for line in fp.readlines():
q, a, doc, b = line.split()

rel_docs.setdefault(q, [])
rel_docs[q].append(doc)

return rel_docs
return {}

def get_relevance_docs(self, q):
return self.rel_docs[q]

def get_result_docs(self):
return self.res_docs

```

```
def build_json_input(self, xml_query_file, id, json_query_file):
json_query = json.dumps(xmltodict.parse(open(xml_query_file).read()))
json_query = json.loads(json_query)
json_query = json_query['parameters']['query']

selected_json_query = {}
selected_json_query.setdefault('query', [])
for query in json_query:
if query['number'] == id:
selected_json_query['query'].append({
'number' : id,
'text' : query['text']
})

open(json_query_file, 'wb').write(json.dumps(selected_json_query))

return selected_json_query['query'][0]

def search(self, index_dir, json_query_file, result_file, count):
res_docs = []

cmd = Command([self.galago_bin, 'batch-search', '--index={}'.format(index_dir),
'--requested={}'.format(count), '{}'.format(json_query_file)],
self.search_result)
cmd.run(60 * 10, args=(res_docs, result_file, ))

return res_docs

def search_result(self, out, res_docs, result_file):
lines = []
if out and hasattr(out, 'readline'):
for line in iter(out.readline, b''):
line = line.strip()
q_id, a, doc, id, score, b = self.search_parse(line)
lines.append((q_id, a, doc, id, score, b))
res_docs.append(doc)

with open(result_file, 'wb') as fp:
fp.write('\n'.join([' '.join(l) for l in lines]))
fp.close()

def search_parse(self, line):
parts = line.split()
if len(parts) == 6:
q_id, a, doc, id, score, b = parts
else:
q_id = parts[0]
a = parts[1]
doc = ' '.join(parts[2:len(parts)-3])
id = parts[len(parts) - 3]
score = parts[len(parts) - 2]
b = parts[len(parts) - 1]

doc = os.path.basename(doc)
doc = os.path.splitext(doc)[0]
return q_id, a, doc, id, score, b

def eval(self, rel_file, res_file, eval_file):
bash = '{} eval --judgments="{}" --baseline="{}" > "{}"'.format(
self.galago_bin, rel_file, res_file, eval_file)

cmd = Command(['bash', '-c', bash])
code = cmd.run(60 * 10)

return code == 0

def get_precision(self, rel_docs, res_docs):
relset = set(rel_docs)
retrset = set(res_docs)

return float(len(relset.intersection(retrset))) / len(retrset)

def get_recall(self, rel_docs, res_docs):
relset = set(rel_docs)
retrset = set(res_docs)

return float(len(relset.intersection(retrset))) / len(relset)

def get_all_precisions(self, rel_docs, res_docs):
rr = []
for i in range(1, len(res_docs) + 1):
rr.append(self.get_precision(rel_docs, res_docs[:i]))

return rr

def get_all_recalls(self, rel_docs, res_docs):
rr = []
for i in range(1, len(res_docs) + 1):
rr.append(self.get_recall(rel_docs, res_docs[:i]))

return rr

def get_map(self, rel_docs, res_docs):
rr = self.get_all_precisions(rel_docs, res_docs)

res = []
for i in range(len(res_docs)):
if res_docs[i] in rel_docs:
res.append(rr[i])

if len(res) == 0:
return 0.0

return float(sum(res)) / len(res)

def get_relevance(self, i, rel_docs, res_docs):
return 1 if res_docs[i] in rel_docs else 0

def get_dcg(self, p, rel_docs, res_docs):
sum = 0
for i in range(2, p + 1):
sum += float(self.get_relevance(i-1, rel_docs, res_docs)) / log(i, 2)
return self.get_relevance(0, rel_docs, res_docs) + sum

def get_idcg(self, p):
sum = 0
for i in range(2, p + 1):
sum += 1 / log(i, 2)
```

```
return 1 + sum

def get_ndcg(self, p, rel_docs, res_docs):
dcg = self.get_dcg(p, rel_docs, res_docs)
idcg = self.get_idcg(p)
return dcg / idcg

def get_reciprocal_rank(self, rel_docs, res_docs):
for i in range(1, len(res_docs) + 1):
if res_docs[i - 1] in rel_docs:
return 1.0 / i
return 0.0
```

Listing 2: Code for galago.py

---

## Question 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:

For this assignment, I randomly choose query no. 6 and no. 12 and calculate the recall and precision values based on the top 10 rank positions. The code to do this can be seen on listing 3. To generate the recall-precision graph, I use python library *matplotlib* [3] based on example from [4].
Figure 3 and figure 4 show the uninterpolated and interpolated recall-precision graphs for query no. 6 and 12. The table of interpolated precision values can be seen in figure

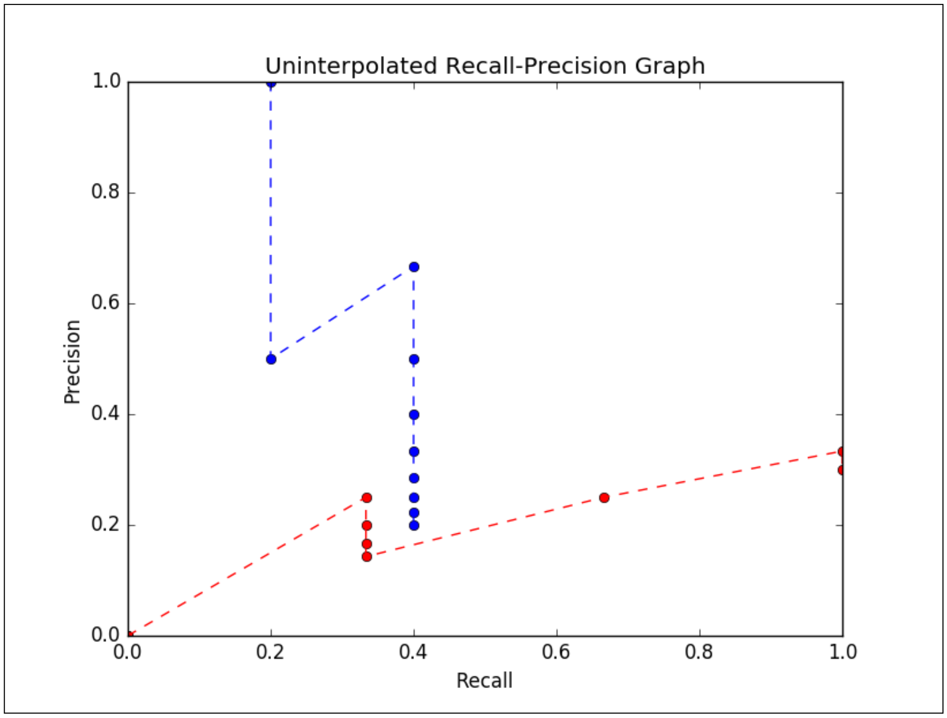


Figure 3: Uninterpolated recall-precision graph for query no. 6 and 12

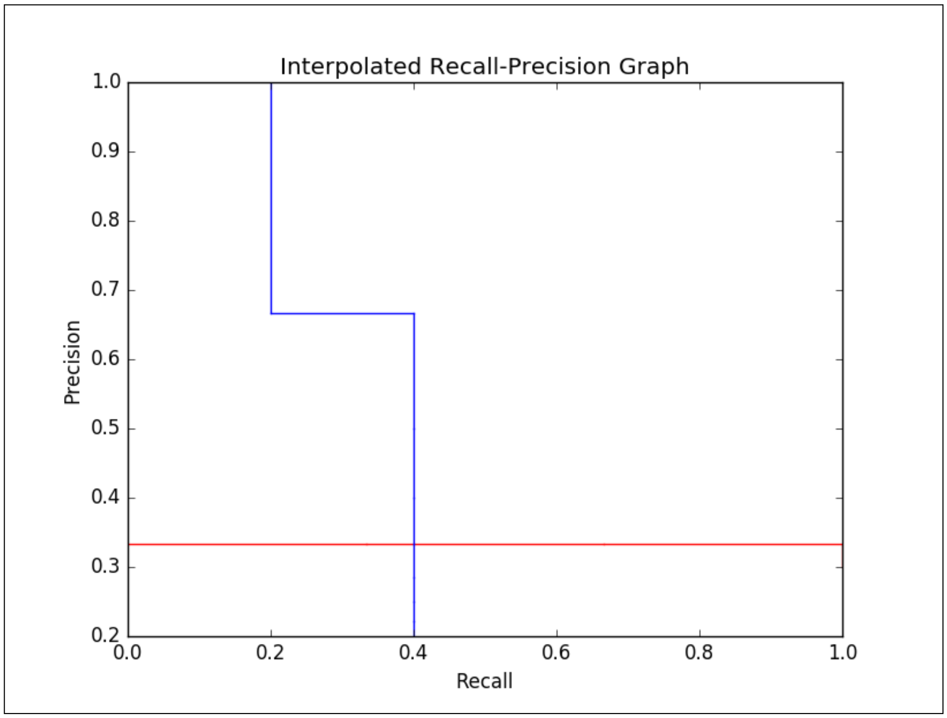
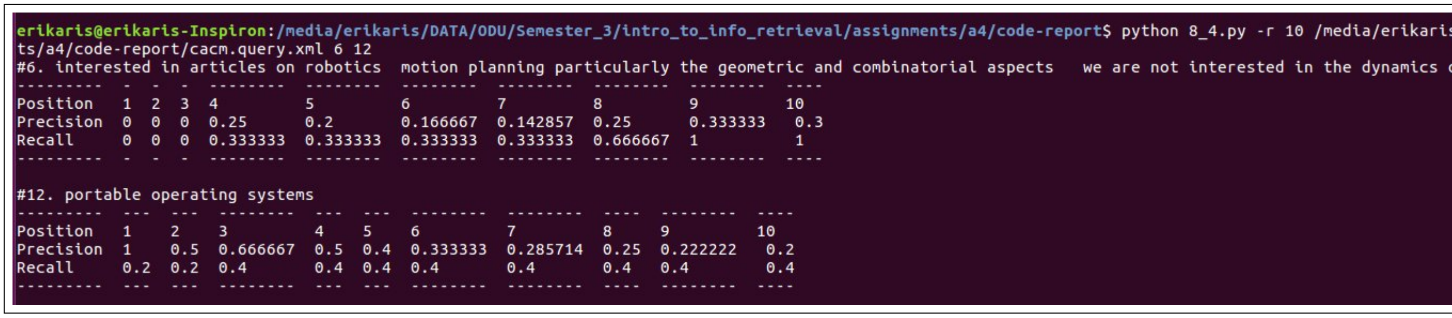


Figure 4: Interpolated recall-precision graph for query no. 6 and 12



```
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report$ python 8_4.py -r 10 /media/erikaris
ts/a4/code-report/cacm.query.xml 6 12
#6. interested in articles on robotics  motion planning particularly the geometric and combinatorial aspects   we are not interested in the dynamics
---------  -  -  -  --------  --------  --------  --------  --------  --------  ----
Position   1  2  3  4         5         6         7         8         9         10
Precision  0  0  0  0.25      0.2       0.166667  0.142857  0.25      0.333333   0.3
Recall     0  0  0  0.333333  0.333333  0.333333  0.333333  0.666667  1          1
---------  -  -  -  --------  --------  --------  --------  --------  --------  ----

#12. portable operating systems
---------  ---  ---  --------  ---  ---  --------  --------  ----  --------  ----
Position   1    2    3         4    5    6         7         8     9         10
Precision  1    0.5  0.666667  0.5  0.4  0.333333  0.285714  0.25  0.222222   0.2
Recall     0.2  0.2  0.4       0.4  0.4  0.4       0.4       0.4   0.4        0.4
---------  ---  ---  --------  ---  ---  --------  --------  ----  --------  ----
```

Figure 5: Table of interpolate for query no. 6 and 12

```
#!/usr/bin/python

import os
from optparse import OptionParser

import errno
from tabulate import tabulate
from lib.galago import GalagoRank
import matplotlib.pyplot as plt
import numpy as np

if __name__ == '__main__':
parser = OptionParser(description='Generate a ranking using Galago')
parser.set_usage(parser.get_usage().replace('\n', '') + ' <xml_file_input> [q1 ... qn]')
parser.add_option('-g', '--galago', dest="galago_bin", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/galago/galago-3.10-bin/bin/galago',
help='Galago "home" directory')
parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cac',
help='Document directory to be indexed')
parser.add_option('-j', '--judgements', dest="judgements_file", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm.rel',
help='File .rel as galago eval judgments')
parser.add_option('-r', '--result', dest="result_count", default='10', help='Number of result')

(options, args) = parser.parse_args()
options = vars(options)

if len(args) < 2:
parser.print_help()
exit()


out_dir = os.path.abspath('output')

try: os.makedirs(out_dir)
except OSError, e:
if e.errno != errno.EEXIST: raise

index_dir = os.path.join(os.path.pardir, 'index')
xml_query_file = os.path.abspath(args[0])

```

```
q_ids = []
if len(args) > 1:
q_ids = args[1:]

recals_precisions = []
galago = GalagoRank(options['galago_bin'], options['judgements_file'])
if galago.index(options['document_dir'], index_dir):
for q_id in q_ids:
json_query_file = os.path.join(out_dir, 'query_{}.json'.format(q_id))
res_file = os.path.join(out_dir, 'result_{}.res'.format(q_id))
eval_file = os.path.join(out_dir, 'result_{}.eval'.format(q_id))

json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)

rel_docs = galago.get_relevance_docs(q_id)
res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count'])

precisions = galago.get_all_precisions(rel_docs, res_docs)
recals = galago.get_all_recalls(rel_docs, res_docs)

recals_precisions.append((recals, precisions))

table = [['Position', ] + range(1, 11)]
table.append(['Precision', ] + precisions)
table.append(['Recall', ] + recals)

print '#{}. {}'.format(json_query['number'], json_query['text'])
print tabulate(table)
print('')


# Uninterpolated
colors = 'rbgcmyk'
for i, (recals, precisions) in enumerate(recals_precisions):
plt.plot(recals, precisions, marker='o', linestyle='None', color=colors[i])
plt.plot(recals, precisions, marker='None', linestyle='--', color=colors[i])

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Uninterpolated Recall-Precision Graph')
plt.savefig(os.path.join(out_dir, 'uninterpolated.png'))
plt.show()
plt.clf()

# Interpolated
# Reference : http://stackoverflow.com/questions/39836953/how-to-draw-a-precision-recall-curve-with-interpolation-in-python
colors = 'rbgcmyk'
for j, (recals, precisions) in enumerate(recals_precisions):
recals = np.asarray(recals)
precisions = np.asarray(precisions)
precisions2 = precisions.copy()

i = recals.shape[0] - 2
while i >= 0:
if precisions[i + 1] > precisions[i]:
precisions[i] = precisions[i + 1]
i = i - 1
```

```

for i in range(recals.shape[0] - 1):
plt.plot((recals[i], recals[i]), (precisions[i], precisions[i + 1]), 'k-', label='', color=colors[j])  # vertical
plt.plot((recals[i], recals[i + 1]), (precisions[i + 1], precisions[i + 1]), 'k-', label='', color=colors[j])  # horizontal
# plt.plot(recals, precisions2, 'k--', color=colors[j])

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Interpolated Recall-Precision Graph')
plt.savefig(os.path.join(out_dir, 'interpolated.png'))
plt.show()
```

Listing 3: Code for question 8.4

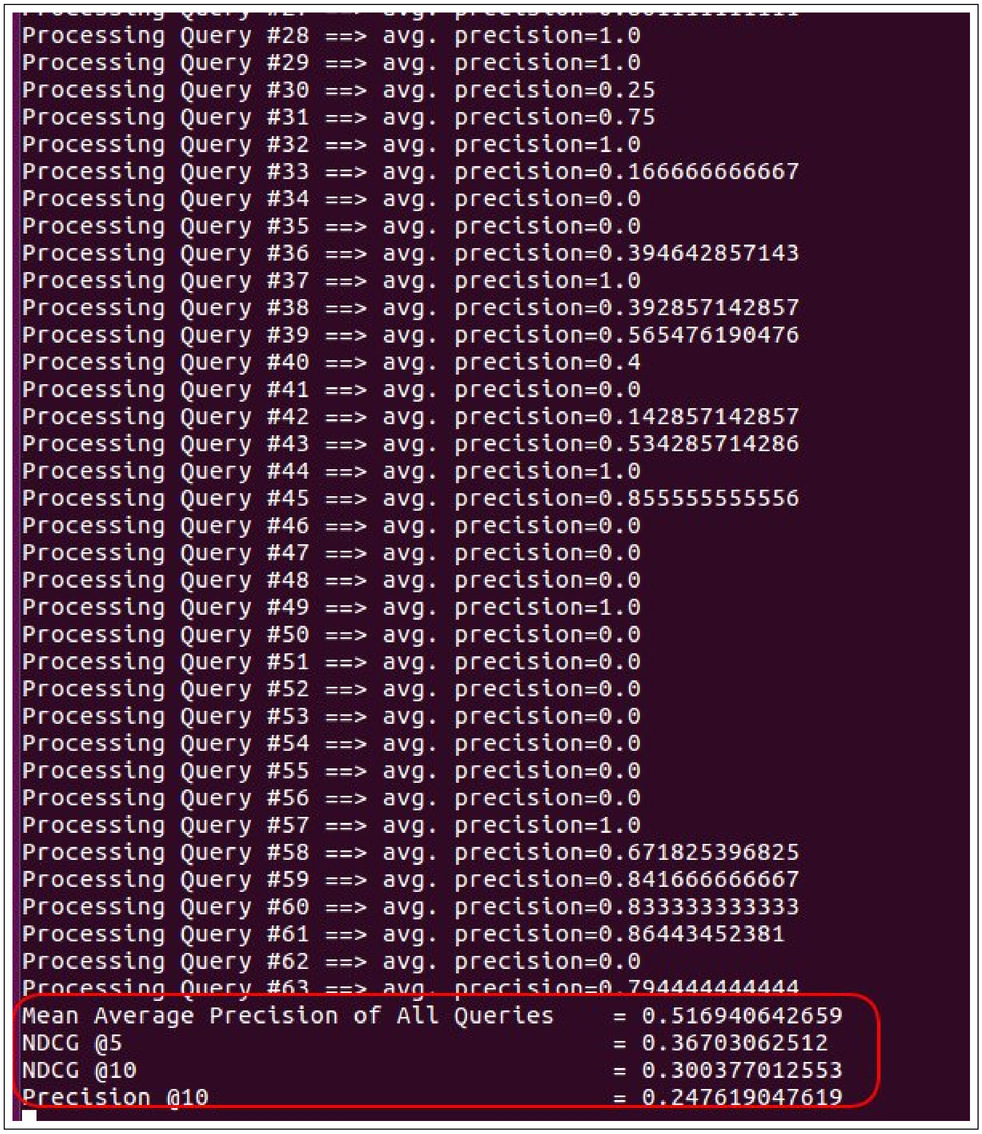
---

## Question 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

The code to solve this problem can be found on listing 4. Figure 6 shows the value of mean average precision (MAP), average NDCG at 5 and 10, and precision at 10 for the entire CACM query set. The recall-precision graph is shown in figure 7.



```
Processing Query #27 ==> avg. precision=0.861111111111
Processing Query #28 ==> avg. precision=1.0
Processing Query #29 ==> avg. precision=1.0
Processing Query #30 ==> avg. precision=0.25
Processing Query #31 ==> avg. precision=0.75
Processing Query #32 ==> avg. precision=1.0
Processing Query #33 ==> avg. precision=0.166666666667
Processing Query #34 ==> avg. precision=0.0
Processing Query #35 ==> avg. precision=0.0
Processing Query #36 ==> avg. precision=0.394642857143
Processing Query #37 ==> avg. precision=1.0
Processing Query #38 ==> avg. precision=0.392857142857
Processing Query #39 ==> avg. precision=0.565476190476
Processing Query #40 ==> avg. precision=0.4
Processing Query #41 ==> avg. precision=0.0
Processing Query #42 ==> avg. precision=0.142857142857
Processing Query #43 ==> avg. precision=0.534285714286
Processing Query #44 ==> avg. precision=1.0
Processing Query #45 ==> avg. precision=0.855555555556
Processing Query #46 ==> avg. precision=0.0
Processing Query #47 ==> avg. precision=0.0
Processing Query #48 ==> avg. precision=0.0
Processing Query #49 ==> avg. precision=1.0
Processing Query #50 ==> avg. precision=0.0
Processing Query #51 ==> avg. precision=0.0
Processing Query #52 ==> avg. precision=0.0
Processing Query #53 ==> avg. precision=0.0
Processing Query #54 ==> avg. precision=0.0
Processing Query #55 ==> avg. precision=0.0
Processing Query #56 ==> avg. precision=0.0
Processing Query #57 ==> avg. precision=1.0
Processing Query #58 ==> avg. precision=0.671825396825
Processing Query #59 ==> avg. precision=0.841666666667
Processing Query #60 ==> avg. precision=0.833333333333
Processing Query #61 ==> avg. precision=0.86443452381
Processing Query #62 ==> avg. precision=0.0
Processing Query #63 ==> avg. precision=0.794444444444
Mean Average Precision of All Queries    = 0.516940642659
NDCG @5                                  = 0.36703062512
NDCG @10                                 = 0.300377012553
Precision @10                            = 0.247619047619
```

Figure 6: MAP, NDCG at 5 and 10, precision at 10 for the entire CACM query set

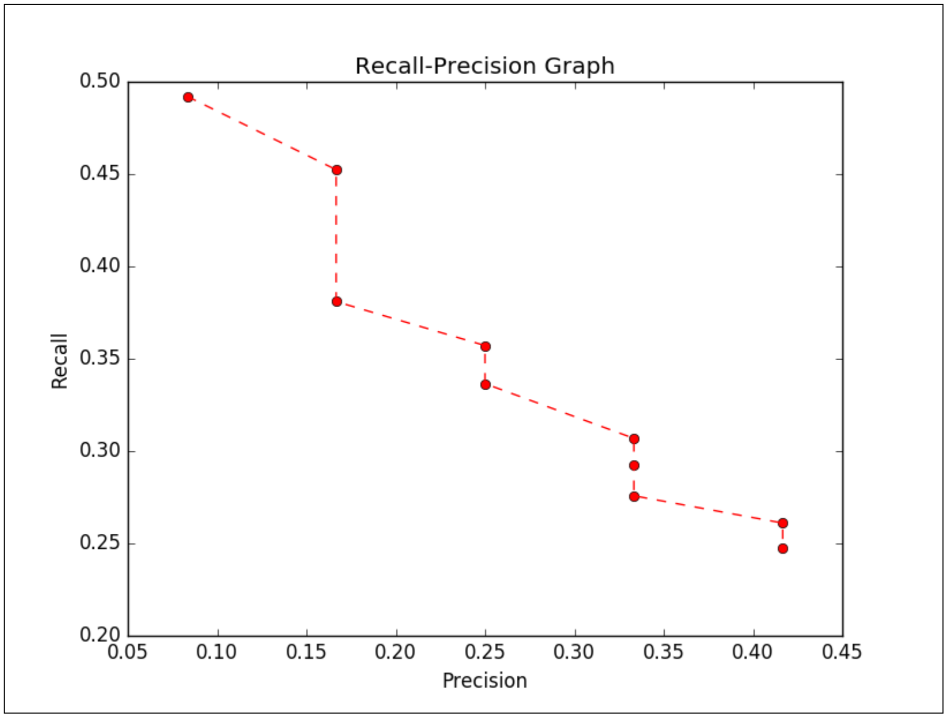


Figure 7: Recall-precision graph for the entire CACM query set

```
#!/usr/bin/python
import errno
import os
from optparse import OptionParser
import matplotlib.pyplot as plt
import numpy
from lib.galago import GalagoRank

if __name__ == '__main__':
parser = OptionParser(description='Generate a ranking using Galago')
parser.set_usage(parser.get_usage().replace('\n', '') + ' <xml_file_input>')
parser.add_option('-g', '--galago', dest="galago_bin", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/galago/galago-3.10-bin/bin/galago',
help='Galago "home" directory')
parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm',
help='Document directory to be indexed')
parser.add_option('-j', '--judgements', dest="judgements_file", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm.rel',
help='File .rel as galago eval judgments')
parser.add_option('-r', '--result', dest="result_count", default='10', help='Number of result')

(options, args) = parser.parse_args()
options = vars(options)

if len(args) < 1:
parser.print_help()
exit()

```

```

out_dir = os.path.abspath('output')

try: os.makedirs(out_dir)
except OSError, e:
if e.errno != errno.EEXIST: raise

index_dir = os.path.abspath(os.path.join(os.path.pardir, 'index'))
xml_query_file = os.path.abspath(args[0])
json_query_file = os.path.join(out_dir, 'query.json')
rel_file = os.path.join(out_dir, 'result.rel')
res_file = os.path.join(out_dir, 'result.res')
eval_file = os.path.join(out_dir, 'result.eval')

recals_precisions = []
maps = []
ndcg_5s = []
ndcg_10s = []
prec_10s = []
galago = GalagoRank(options['galago_bin'], options['judgements_file'])
if galago.index(options['document_dir'], index_dir):
for q_id in range(1, 64):
json_query_file = os.path.join(out_dir, 'query_{}.json'.format(q_id))
res_file = os.path.join(out_dir, 'result_{}.res'.format(q_id))
eval_file = os.path.join(out_dir, 'result_{}.eval'.format(q_id))

json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)

rel_docs = galago.get_relevance_docs(q_id)
res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count'
    ])

precisions = galago.get_all_precisions(rel_docs, res_docs)
recals = galago.get_all_recalls(rel_docs, res_docs)

map = galago.get_map(rel_docs, res_docs)
maps.append(map)
ndcg_5s.append(galago.get_ndcg(5, rel_docs, res_docs))
ndcg_10s.append(galago.get_ndcg(10, rel_docs, res_docs))
prec_10s.append(galago.get_all_precisions(rel_docs, res_docs)[9])

recals_precisions.append((recals, precisions))

print 'Processing Query #{} ==> avg. precision={}'.format(q_id, map)

# Calculate avg of map
print 'Mean Average Precision of All Queries    = {}'.format(float(sum(maps)) / len(
    maps))
print 'NDCG @5                                  = {}'.format(float(sum(ndcg_5s)) /
    len(ndcg_5s))
print 'NDCG @10                                 = {}'.format(float(sum(ndcg_10s)) /
    len(ndcg_10s))
print 'Precision @10                            = {}'.format(float(sum(prec_10s)) /
    len(prec_10s))

# Graph
# Transpose
recals_precisions = numpy.asarray(recals_precisions).T.tolist()

```

```
recalls = []
precisions = []
for d_recalls, d_precisions in recals_precisions:
recalls.append(float(sum(d_recalls)) / len(d_recalls))
precisions.append(float(sum(d_precisions)) / len(d_precisions))

plt.plot(recals, precisions, marker='o', linestyle='None', color='r')
plt.plot(recals, precisions, marker='None', linestyle='--', color='r')

plt.xlabel('Precision')
plt.ylabel('Recall')
plt.title('Recall-Precision Graph')
plt.savefig(os.path.join(out_dir, 'recall-precision.png'))
plt.show()
plt.clf()
```

Listing 4: Code for question 8.5

## Question 8.7

Another measure that has been used in a number of evaluations isR-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

## Question 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

For this question, I choose query no. 3 and calculate its BPREF values. BPREF value can be calculated using 2 formulas:

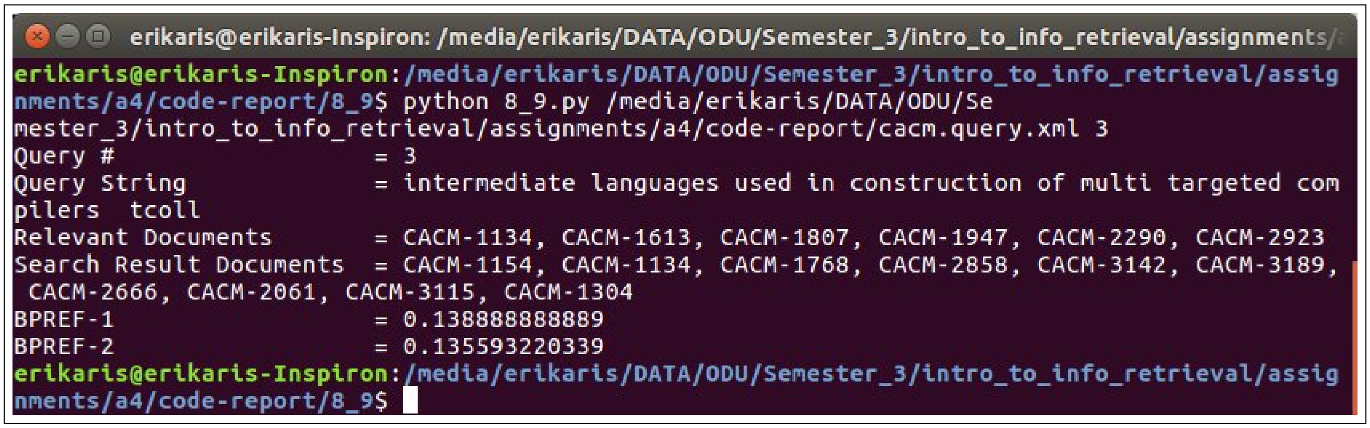
$$BPREF = \frac{1}{R}\sum_{d_r}(1 - \frac{N_{d_r}}{R}) \tag{5}$$

and

$$BPREF = \frac{P}{P+Q} \tag{6}$$

$N_{d_r}$ is the number of non-relevant documents that are ranked higher than the relevant document $d_r$. R is the number of relevant documents. P is the number of preferences that agree with the rank and Q is the number of preferences that disagree with the rank.

Figure 8 shows the result of BPREF calculation for query no. 3.



```
erikaris@erikaris-Inspiron: /media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assig
nments/a4/code-report/8_9$ python 8_9.py /media/erikaris/DATA/ODU/Se
mester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm.query.xml 3
Query #                    = 3
Query String               = intermediate languages used in construction of multi targeted com
pilers  tcoll
Relevant Documents         = CACM-1134, CACM-1613, CACM-1807, CACM-1947, CACM-2290, CACM-2923
Search Result Documents    = CACM-1154, CACM-1134, CACM-1768, CACM-2858, CACM-3142, CACM-3189,
 CACM-2666, CACM-2061, CACM-3115, CACM-1304
BPREF-1                    = 0.138888888889
BPREF-2                    = 0.135593220339
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assig
nments/a4/code-report/8_9$
```

Figure 8: BPREF values for query no. 3

From figure 8, we can see that the 2 formulations of BPREF values give the similar value that are only slight different in the number of decimal places. The code to generate the BPREF values can be found on listing 5

```
#!/usr/bin/python

import errno
import os
from optparse import OptionParser
from lib.galago import GalagoRank

if __name__ == '__main__':
parser = OptionParser(description='Generate a ranking using Galago')
parser.set_usage(parser.get_usage().replace('\n', '') + ' <xml_file_input> [q1 ... qn]')
parser.add_option('-g', '--galago', dest="galago_bin", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/galago/galago-3.10-bin/bin/galago',
help='Galago "home" directory')
parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm',
help='Document directory to be indexed')
parser.add_option('-j', '--judgements', dest="judgements_file", default='/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm.rel',
help='File .rel as galago eval judgments')
parser.add_option('-r', '--result', dest="result_count", default='10', help='Number of result')

(options, args) = parser.parse_args()
options = vars(options)

if len(args) < 2:
parser.print_help()
exit()


out_dir = os.path.abspath('output')

try: os.makedirs(out_dir)
```

```
except OSError, e:
if e.errno != errno.EEXIST: raise

index_dir = os.path.join(os.path.pardir, 'index')
xml_query_file = os.path.abspath(args[0])
json_query_file = os.path.join(out_dir, 'query.json')
rel_file = os.path.join(out_dir, 'result.rel')
res_file = os.path.join(out_dir, 'result.res')
eval_file = os.path.join(out_dir, 'result.eval')

q_ids = []
if len(args) > 1:
q_ids = args[1:]

galago = GalagoRank(options['galago_bin'], options['judgements_file'])
if galago.index(options['document_dir'], index_dir):
for q_id in q_ids:
json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)

rel_docs = galago.get_relevance_docs(q_id)
res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count']
    ])

galago.eval(options['judgements_file'], res_file, eval_file)

print 'Query #                  = {}'.format(json_query['number'])
print 'Query String             = {}'.format(json_query['text'])
print 'Relevant Documents       = {}'.format(', '.join(rel_docs))
print 'Search Result Documents  = {}'.format(', '.join(res_docs))
print 'BPREF-1                  = {}'.format(galago.get_bpref_1(rel_docs, res_docs))
print 'BPREF-2                  = {}'.format(galago.get_bpref_2(rel_docs, res_docs))
```

Listing 5: Code for calculating BPREF values

# References

[1] The Lemur Project. The Lemur Project - Galago 3.10 . https://sourceforge.net/p/lemur/galago/ci/release-3.10/tree/, 2016. [Online; accessed 5-November-2016].

[2] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 1st edition, 2009.

[3] The Matplotlib development team. Matplotlib . http://matplotlib.org/, 2016. [Online; accessed 7-December-2016].

[4] Stackoverflow. How to draw a precision-recall curve with interpolation in python? . http://stackoverflow.com/questions/39836953/how-to-draw-a-precision-recall-curve-with-interpolation-in-python, 2016. [Online; accessed 7-December-2016].