

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.

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

$$Recall = \frac{|A \cap B|}{|A|} \quad (1)$$

$$Precision = \frac{|A \cap B|}{|B|} \quad (2)$$

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:

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

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:

$$NDCG_p = \frac{DCG_p}{IDCG_p} \quad (4)$$

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

```
1 #!/usr/bin/python
2
3 import errno
4 import os
5 from optparse import OptionParser
6 from lib.galago import GalagoRank
7
8 if __name__ == '__main__':
9     parser = OptionParser(description='Generate a ranking using Galago')
10    parser.set_usage(parser.get_usage().replace('\n', ' ') + ' <xml_file_input> [q1 ... qn
    ]')
11    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',
12    help='Galago "home" directory')
13    parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/
    DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm',
14    help='Document directory to be indexed')
15    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',
16    help='File .rel as galago eval judgments')
17    parser.add_option('-r', '--result', dest="result_count", default='10', help='Number
    of result')
18
19    (options, args) = parser.parse_args()
20    options = vars(options)
21
22    if len(args) < 2:
23        parser.print_help()
24        exit()
25
26
27    out_dir = os.path.abspath('output')
28
29    try: os.makedirs(out_dir)
30    except OSError, e:
31        if e.errno != errno.EEXIST: raise
32
33    index_dir = os.path.join(os.path.pardir, 'index')
```

```

34 xml_query_file = os.path.abspath(args[0])
35 json_query_file = os.path.join(out_dir, 'query.json')
36 rel_file = os.path.join(out_dir, 'result.rel')
37 res_file = os.path.join(out_dir, 'result.res')
38 eval_file = os.path.join(out_dir, 'result.eval')
39
40 q_ids = []
41 if len(args) > 1:
42     q_ids = args[1:]
43
44 galago = GalagoRank(options['galago_bin'], options['judgements_file'])
45 if galago.index(options['document_dir'], index_dir):
46     for q_id in q_ids:
47         json_query = galago.build_json_input(xml_query_file, q_id, json_query_file)
48
49         rel_docs = galago.get_relevance_docs(q_id)
50         res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count',
51                                     ])
52
53         galago.eval(options['judgements_file'], res_file, eval_file)
54
55         print 'Query #                = {}'.format(json_query['number'])
56         print 'Query String           = {}'.format(json_query['text'])
57         print 'Relevant Documents     = {}'.format(', '.join(rel_docs))
58         print 'Search Result Documents = {}'.format(', '.join(res_docs))
59         print 'Average Precision      = {}'.format(galago.get_map(rel_docs, res_docs))
60         print 'NDCG @5                = {}'.format(galago.get_ndcg(5, rel_docs, res_docs))
61         print 'NDCG @10               = {}'.format(galago.get_ndcg(10, rel_docs, res_docs))
62         print 'Precision @10          = {}'.format(galago.get_all_precisions(rel_docs,
63                                     res_docs)[9])
64         print 'Reciprocal Rank        = {}'.format(galago.get_reciprocal_rank(rel_docs,
65                                     res_docs))

```

Listing 1: Code for question 8.3

```

1  #!/usr/bin/python
2
3  import json
4  import os
5  from math import log
6  from subprocess import Popen, PIPE, call
7  from threading import Thread
8  import xmltodict
9
10
11 class Command(object):
12     def __init__(self, cmd, out_pipe_callback=None, err_pipe_callback=None):
13         self.cmd = cmd
14         self.process = None
15         self.out_pipe_callback = out_pipe_callback
16         self.err_pipe_callback = err_pipe_callback
17
18     def run(self, timeout, args=()):
19         def target():
20             self.process = Popen(self.cmd, stdout=PIPE, stderr=PIPE)
21
22         if self.out_pipe_callback:
23             stdout_thread = Thread(target=target,
24                                     args=(self.process.stdout, ) + args)
25             stdout_thread.daemon = True

```

```

26 stdout_thread.start()
27
28 if self.err_pipe_callback:
29 stderr_thread = Thread(target=self.err_pipe_callback,
30 args=(self.process.stderr, ) + args)
31 stderr_thread.daemon = True
32 stderr_thread.start()
33
34 self.process.wait()
35
36 thread = Thread(target=target)
37 thread.daemon = True
38 thread.start()
39
40 thread.join(timeout)
41 try: self.process.terminate()
42 except: pass
43
44 return self.process.returncode
45
46
47 class GalagoRank(object):
48 galago_bin = None
49 rel_docs = {}
50 res_docs = []
51
52 def __init__(self, galago_bin, judgements_file):
53 self.galago_bin = galago_bin
54 self.rel_docs = self.build_relevance(judgements_file)
55
56 def index(self, document_dir, index_dir):
57 if not os.path.exists(index_dir):
58 bash = '"{}" build --indexPath={} --inputPath={} '.format(
59 self.galago_bin, index_dir, document_dir)
60
61 code = call(['bash', '-c', bash])
62 return code == 0
63
64 else:
65 return True
66
67 def build_relevance(self, judgements_file):
68 with open(judgements_file, 'r') as fp:
69 rel_docs = {}
70 for line in fp.readlines():
71 q, a, doc, b = line.split()
72
73 rel_docs.setdefault(q, [])
74 rel_docs[q].append(doc)
75
76 return rel_docs
77 return {}
78
79 def get_relevance_docs(self, q):
80 return self.rel_docs[q]
81
82 def get_result_docs(self):
83 return self.res_docs
84

```

```

85 def build_json_input(self, xml_query_file, id, json_query_file):
86     json_query = json.dumps(xmltodict.parse(open(xml_query_file).read()))
87     json_query = json.loads(json_query)
88     json_query = json_query['parameters']['query']
89
90     selected_json_query = {}
91     selected_json_query.setdefault('query', [])
92     for query in json_query:
93         if query['number'] == id:
94             selected_json_query['query'].append({
95                 'number': id,
96                 'text': query['text']
97             })
98
99     open(json_query_file, 'wb').write(json.dumps(selected_json_query))
100
101     return selected_json_query['query'][0]
102
103 def search(self, index_dir, json_query_file, result_file, count):
104     res_docs = []
105
106     cmd = Command([self.galago_bin, 'batch-search', '--index={}'.format(index_dir),
107                    '--requested={}'.format(count), '{}'.format(json_query_file)],
108                  self.search_result)
109     cmd.run(60 * 10, args=(res_docs, result_file, ))
110
111     return res_docs
112
113 def search_result(self, out, res_docs, result_file):
114     lines = []
115     if out and hasattr(out, 'readline'):
116         for line in iter(out.readline, b''):
117             line = line.strip()
118             q_id, a, doc, id, score, b = self.search_parse(line)
119             lines.append((q_id, a, doc, id, score, b))
120             res_docs.append(doc)
121
122     with open(result_file, 'wb') as fp:
123         fp.write('\n'.join([' '.join(l) for l in lines]))
124         fp.close()
125
126 def search_parse(self, line):
127     parts = line.split()
128     if len(parts) == 6:
129         q_id, a, doc, id, score, b = parts
130     else:
131         q_id = parts[0]
132         a = parts[1]
133         doc = ' '.join(parts[2:len(parts)-3])
134         id = parts[len(parts) - 3]
135         score = parts[len(parts) - 2]
136         b = parts[len(parts) - 1]
137
138     doc = os.path.basename(doc)
139     doc = os.path.splitext(doc)[0]
140     return q_id, a, doc, id, score, b
141
142 def eval(self, rel_file, res_file, eval_file):
143     bash = '{} eval --judgments={}{} --baseline={}{} > {}'.format(

```

```

144 self.galago_bin, rel_file, res_file, eval_file)
145
146 cmd = Command(['bash', '-c', bash])
147 code = cmd.run(60 * 10)
148
149 return code == 0
150
151 def get_precision(self, rel_docs, res_docs):
152     relset = set(rel_docs)
153     retrset = set(res_docs)
154
155     return float(len(relset.intersection(retrset))) / len(retrset)
156
157 def get_recall(self, rel_docs, res_docs):
158     relset = set(rel_docs)
159     retrset = set(res_docs)
160
161     return float(len(relset.intersection(retrset))) / len(relset)
162
163 def get_all_precisions(self, rel_docs, res_docs):
164     rr = []
165     for i in range(1, len(res_docs) + 1):
166         rr.append(self.get_precision(rel_docs, res_docs[:i]))
167
168     return rr
169
170 def get_all_recalls(self, rel_docs, res_docs):
171     rr = []
172     for i in range(1, len(res_docs) + 1):
173         rr.append(self.get_recall(rel_docs, res_docs[:i]))
174
175     return rr
176
177 def get_map(self, rel_docs, res_docs):
178     rr = self.get_all_precisions(rel_docs, res_docs)
179
180     res = []
181     for i in range(len(res_docs)):
182         if res_docs[i] in rel_docs:
183             res.append(rr[i])
184
185     if len(res) == 0:
186         return 0.0
187
188     return float(sum(res)) / len(res)
189
190 def get_relevance(self, i, rel_docs, res_docs):
191     return 1 if res_docs[i] in rel_docs else 0
192
193 def get_dcg(self, p, rel_docs, res_docs):
194     sum = 0
195     for i in range(2, p + 1):
196         sum += float(self.get_relevance(i-1, rel_docs, res_docs)) / log(i, 2)
197     return self.get_relevance(0, rel_docs, res_docs) + sum
198
199 def get_idcg(self, p):
200     sum = 0
201     for i in range(2, p + 1):
202         sum += 1 / log(i, 2)

```



```

203 return 1 + sum
204
205 def get_ndcg(self, p, rel_docs, res_docs):
206     dcg = self.get_dcg(p, rel_docs, res_docs)
207     idcg = self.get_idcg(p)
208     return dcg / idcg
209
210 def get_reciprocal_rank(self, rel_docs, res_docs):
211     for i in range(1, len(res_docs) + 1):
212         if res_docs[i - 1] in rel_docs:
213             return 1.0 / i
214     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, while figure 6 shows the average interpolated recall-precision graph for those 2 queries. The table of interpolated precision values can be seen in figure 5

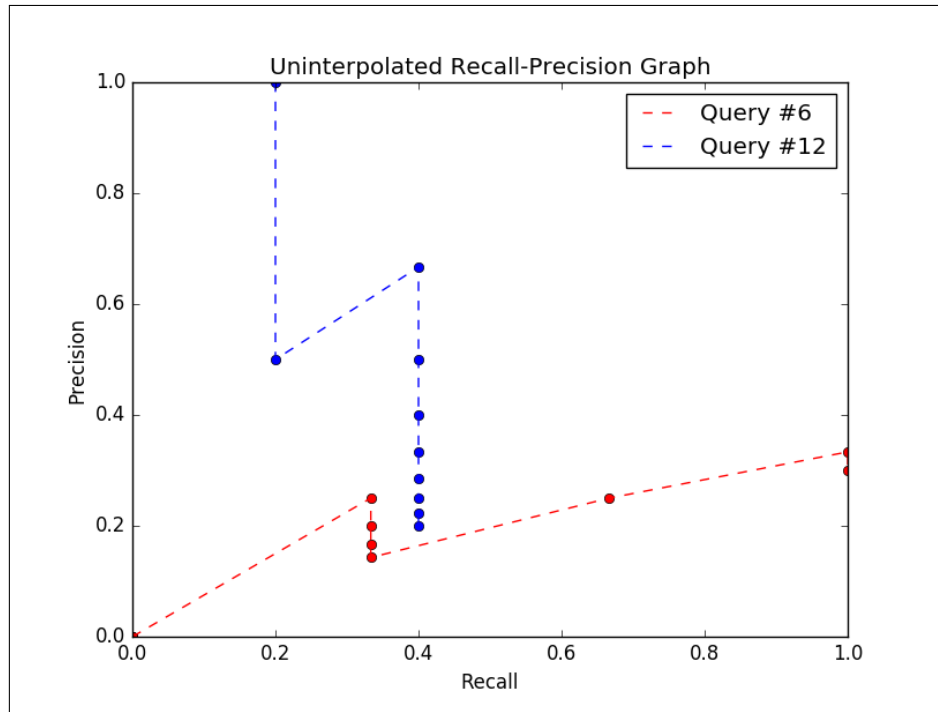


Figure 3: Uninterpolated recall-precision graph for query no. 6 and 12

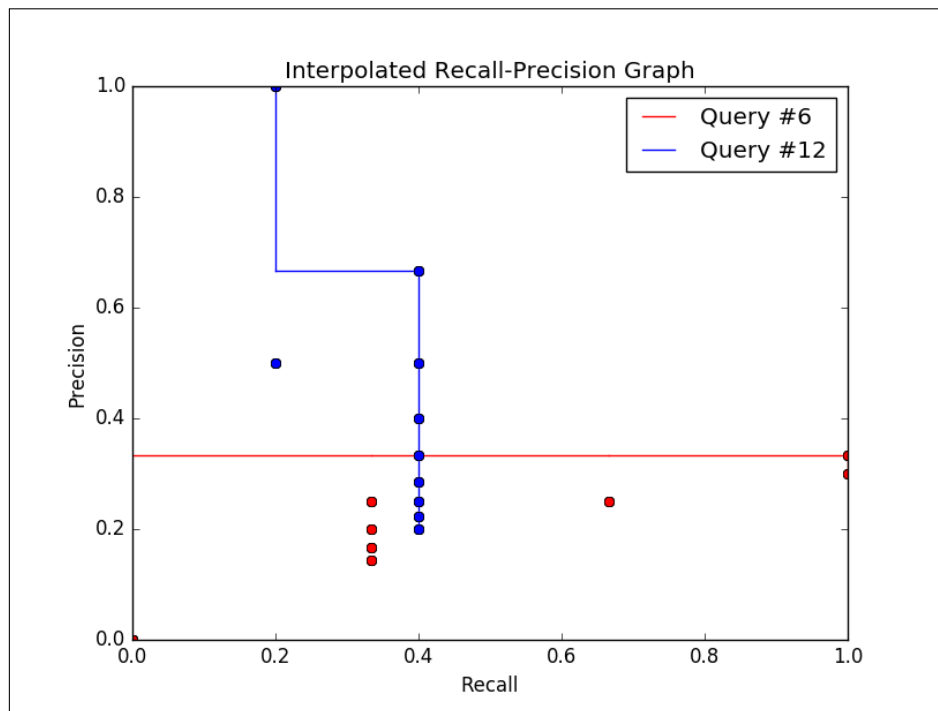


Figure 4: Interpolated recall-precision graph for query no. 6 and 12

```

erikaris@erikaris-Inspiron: /media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/8_4$ py
ro_to_info_retrieval/assignments/a4/code-report/cacm.query.xml 6 12
#6. interested in articles on robotics motion planning particularly the geometric and combinatorial aspects
-----
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
-----

Interpolated Precision
-----
Recall     0          0.1      0.2      0.3      0.4      0.5      0.6      0.7      0.8      0.9
Query #6   0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.3
Query #12  0.666667  0.666667  0.666667  0.5      0.4      0.333333  0.285714  0.25      0.222222  0.2
-----

Average Interpolated Precision
-----
Recall     0          0.1      0.2      0.3      0.4      0.5      0.6      0.7      0.8      0.9
Query #6   0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.333333  0.3
Query #12  0.666667  0.666667  0.666667  0.5      0.4      0.333333  0.285714  0.25      0.222222  0.2
Average    0.5      0.5      0.5      0.416667  0.366667  0.333333  0.309524  0.291667  0.277778  0.25
-----

```

Figure 5: Interpolated precision values for query 6 and 12

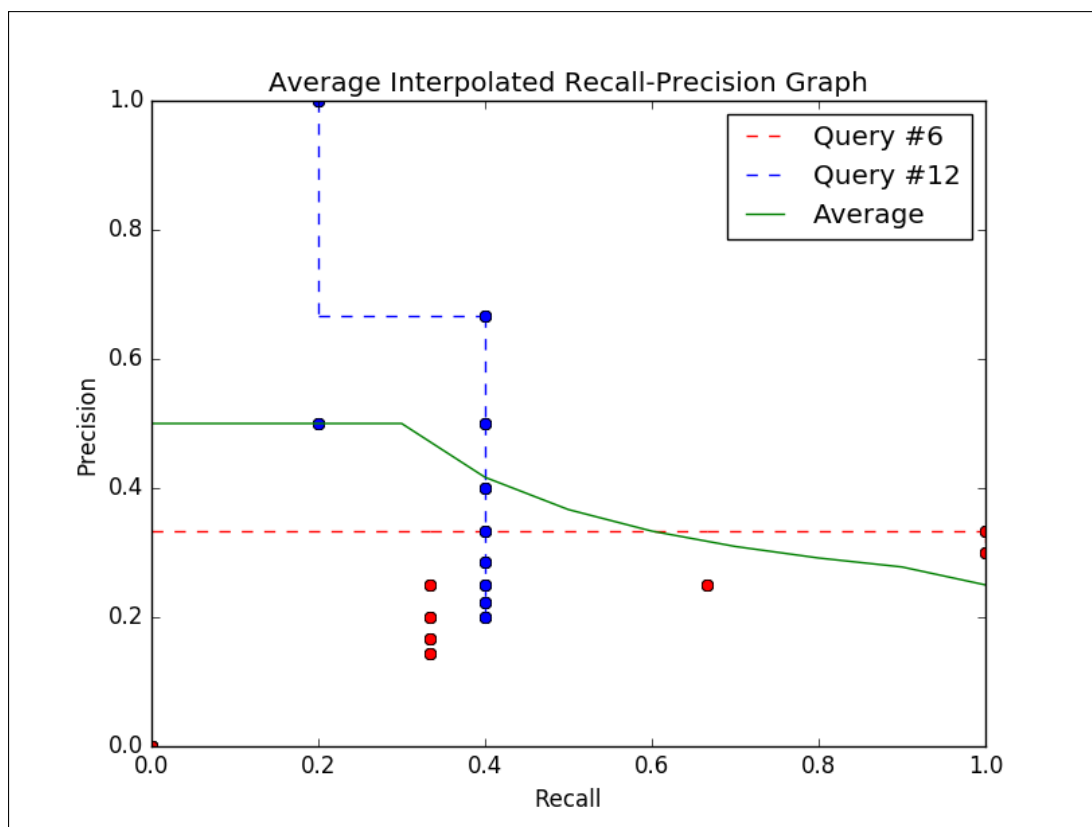


Figure 6: The average interpolated recall-precision graph for query no. 6 and 12

```

1  #!/usr/bin/python
2
3  import os
4  from optparse import OptionParser
5
6  import errno
7  from tabulate import tabulate
8  from lib.galago import GalagoRank
9  import matplotlib.pyplot as plt
10 import numpy as np
11
12 if __name__ == '__main__':
13     parser = OptionParser(description='Generate a ranking using Galago')
14     parser.set_usage(parser.get_usage().replace('\n', ' ') + ' <xml_file_input> [q1 ... qn
15         ]')
16     parser.add_option('-g', '--galago', dest="galago_bin", default='/media/erikaris/DATA/
17         ODU/Semester_3/intro_to_info_retrieval/galago/galago-3.10-bin/bin/galago',
18         help='Galago "home" directory')
19     parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/
20         DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cac',
21         help='Document directory to be indexed')
22     parser.add_option('-j', '--judgements', dest="judgements_file", default='/media/
23         erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/
24         cacm.rel',
25         help='File .rel as galago eval judgments')
26     parser.add_option('-r', '--result', dest="result_count", default='10', help='Number
27         of result')
28
29     (options, args) = parser.parse_args()
30     options = vars(options)
31
32     if len(args) < 2:
33         parser.print_help()
34         exit()
35
36     out_dir = os.path.abspath('output')
37
38     try: os.makedirs(out_dir)
39     except OSError, e:
40         if e.errno != errno.EEXIST: raise
41
42     index_dir = os.path.join(os.path.pardir, 'index')
43     xml_query_file = os.path.abspath(args[0])
44
45     q_ids = []
46     if len(args) > 1:
47         q_ids = args[1:]
48
49     recals_precisions = []
50     galago = GalagoRank(options['galago_bin'], options['judgements_file'])
51     if galago.index(options['document_dir'], index_dir):
52         for q_id in q_ids:
53             json_query_file = os.path.join(out_dir, 'query_{}.json'.format(q_id))
54             res_file = os.path.join(out_dir, 'result_{}.res'.format(q_id))
55             eval_file = os.path.join(out_dir, 'result_{}.eval'.format(q_id))
56
57             json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)

```

```

54 rel_docs = galago.get_relevance_docs(q_id)
55 res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count'
    ])
56
57 precisions = galago.get_all_precisions(rel_docs, res_docs)
58 recals = galago.get_all_recalls(rel_docs, res_docs)
59
60 recals_precisions.append((recals, precisions))
61
62 table = [['Position', ] + range(1, 11)]
63 table.append(['Precision', ] + precisions)
64 table.append(['Recall', ] + recals)
65
66 print '#{. {} '.format(json_query['number'], json_query['text'])
67 print tabulate(table)
68 print('')
69
70
71 # Uninterpolated
72 colors = 'rbgcmk'
73 for i, (recals, precisions) in enumerate(recals_precisions):
74     plt.plot(recals, precisions, marker='o', linestyle=None, color=colors[i])
75     plt.plot(recals, precisions, marker=None, linestyle='—', color=colors[i])
76
77 plt.xlabel('Recall')
78 plt.ylabel('Precision')
79 plt.title('Uninterpolated Recall-Precision Graph')
80 plt.savefig(os.path.join(out_dir, 'uninterpolated.png'))
81 plt.show()
82 plt.clf()
83
84 # Interpolated
85 # Reference : http://stackoverflow.com/questions/39836953/how-to-draw-a-precision-
    recall-curve-with-interpolation-in-python
86 colors = 'rbgcmk'
87 for j, (recals, precisions) in enumerate(recals_precisions):
88     recals = np.asarray(recals)
89     precisions = np.asarray(precisions)
90     precisions2 = precisions.copy()
91
92     i = recals.shape[0] - 2
93     while i >= 0:
94         if precisions[i + 1] > precisions[i]:
95             precisions[i] = precisions[i + 1]
96             i = i - 1
97
98     for i in range(recals.shape[0] - 1):
99         plt.plot((recals[i], recals[i]), (precisions[i], precisions[i + 1]), 'k-', label='',
            color=colors[j]) # vertical
100        plt.plot((recals[i], recals[i + 1]), (precisions[i + 1], precisions[i + 1]), 'k-',
            label='', color=colors[j]) # horizontal
101    # plt.plot(recals, precisions2, 'k--', color=colors[j])
102
103    plt.xlabel('Recall')
104    plt.ylabel('Precision')
105    plt.title('Interpolated Recall-Precision Graph')
106    plt.savefig(os.path.join(out_dir, 'interpolated.png'))
107    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 formulas to calculate the metrics required in this question are similar to the formulas that have been used in the previous questions. The code to solve this problem can be found on listing 4. Figure 7 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 8.

```
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 7: MAP, NDCG at 5 and 10, precision at 10 for the entire CACM query set

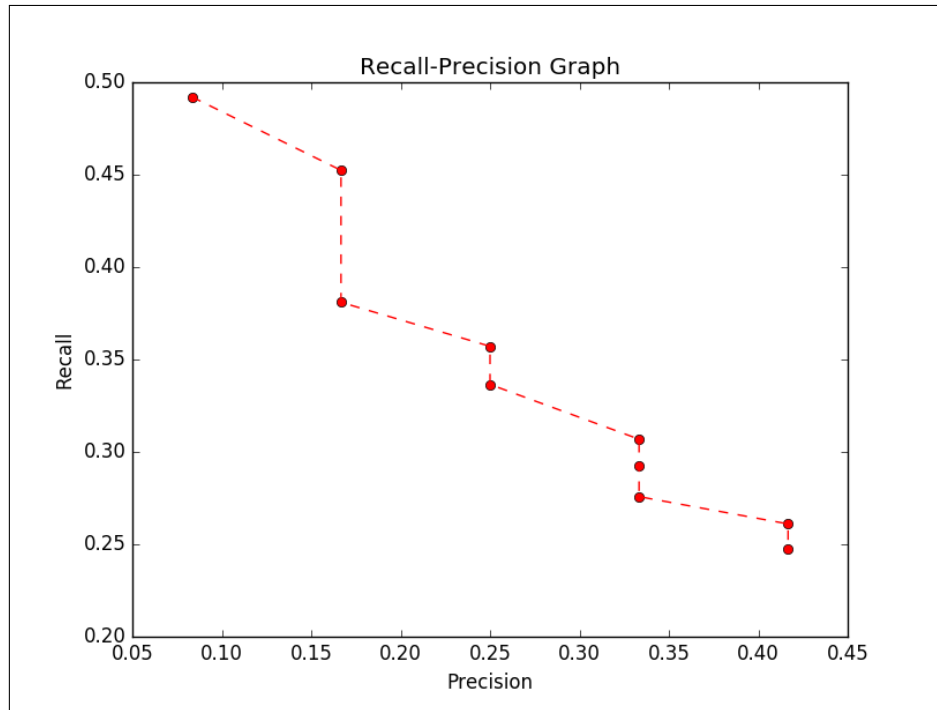


Figure 8: Recall-precision graph for the entire CACM query set

```

1  #!/usr/bin/python
2  import errno
3  import os
4  from optparse import OptionParser
5  import matplotlib.pyplot as plt
6  import numpy
7  from lib.galago import GalagoRank
8
9  if __name__ == '__main__':
10 parser = OptionParser(description='Generate a ranking using Galago')
11 parser.set_usage(parser.get_usage().replace('\n', ' ') + ' <xml_file_input>')
12 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',
13 help='Galago "home" directory')
14 parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/
    DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm',
15 help='Document directory to be indexed')
16 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',
17 help='File .rel as galago eval judgments')
18 parser.add_option('-r', '--result', dest="result_count", default='10', help='Number
    of result')
19
20 (options, args) = parser.parse_args()
21 options = vars(options)
22
23 if len(args) < 1:
24 parser.print_help()
25 exit()
26

```

```

27
28 out_dir = os.path.abspath('output')
29
30 try: os.makedirs(out_dir)
31 except OSError, e:
32     if e.errno != errno.EEXIST: raise
33
34 index_dir = os.path.abspath(os.path.join(os.path.pardir, 'index'))
35 xml_query_file = os.path.abspath(args[0])
36 json_query_file = os.path.join(out_dir, 'query.json')
37 rel_file = os.path.join(out_dir, 'result.rel')
38 res_file = os.path.join(out_dir, 'result.res')
39 eval_file = os.path.join(out_dir, 'result.eval')
40
41 recals_precisions = []
42 maps = []
43 ndcg_5s = []
44 ndcg_10s = []
45 prec_10s = []
46 galago = GalagoRank(options['galago_bin'], options['judgements_file'])
47 if galago.index(options['document_dir'], index_dir):
48     for q_id in range(1, 64):
49         json_query_file = os.path.join(out_dir, 'query_{}.json'.format(q_id))
50         res_file = os.path.join(out_dir, 'result_{}.res'.format(q_id))
51         eval_file = os.path.join(out_dir, 'result_{}.eval'.format(q_id))
52
53         json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)
54
55         rel_docs = galago.get_relevance_docs(q_id)
56         res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count',
57                                     ])
58
59         precisions = galago.get_all_precisions(rel_docs, res_docs)
60         recals = galago.get_all_recalls(rel_docs, res_docs)
61
62         map = galago.get_map(rel_docs, res_docs)
63         maps.append(map)
64         ndcg_5s.append(galago.get_ndcg(5, rel_docs, res_docs))
65         ndcg_10s.append(galago.get_ndcg(10, rel_docs, res_docs))
66         prec_10s.append(galago.get_all_precisions(rel_docs, res_docs)[9])
67
68         recals_precisions.append((recals, precisions))
69
70     print 'Processing Query #{} ==> avg. precision={}'.format(q_id, map)
71
72 # Calculate avg of map
73 print 'Mean Average Precision of All Queries = {}'.format(float(sum(maps)) / len(
74     maps))
75 print 'NDCG @5 = {}'.format(float(sum(ndcg_5s)) /
76     len(ndcg_5s))
77 print 'NDCG @10 = {}'.format(float(sum(ndcg_10s)) /
78     len(ndcg_10s))
79 print 'Precision @10 = {}'.format(float(sum(prec_10s)) /
80     len(prec_10s))
81
82 # Graph
83 # Transpose
84 recals_precisions = numpy.asarray(recals_precisions).T.tolist()
85
86

```



```

81 recalls = []
82 precisions = []
83 for d_recalls, d_precisions in recals_precisions:
84     recalls.append(float(sum(d_recalls)) / len(d_recalls))
85     precisions.append(float(sum(d_precisions)) / len(d_precisions))
86
87 plt.plot(recalls, precisions, marker='o', linestyle='None', color='r')
88 plt.plot(recalls, precisions, marker='None', linestyle='—', color='r')
89
90 plt.xlabel('Precision')
91 plt.ylabel('Recall')
92 plt.title('Recall-Precision Graph')
93 plt.savefig(os.path.join(out_dir, 'recall-precision.png'))
94 plt.show()
95 plt.clf()

```

Listing 4: Code for question 8.5

Question 8.7

Another measure that has been used in a 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

The script to calculate R-precision can be seen in listing 5. Figure 9 shows the comparison between R-precision and other evaluation metrics for the entire CACM query set.

```

Processing Query #36
Processing Query #37
Processing Query #38
Processing Query #39
Processing Query #40
Processing Query #41
Processing Query #42
Processing Query #43
Processing Query #44
Processing Query #45
Processing Query #46
Processing Query #47
Processing Query #48
Processing Query #49
Processing Query #50
Processing Query #51
Processing Query #52
Processing Query #53
Processing Query #54
Processing Query #55
Processing Query #56
Processing Query #57
Processing Query #58
Processing Query #59
Processing Query #60
Processing Query #61
Processing Query #62
Processing Query #63
Mean Average Precision    = 0.473091913278
NDCG @5                  = 0.36703062512
NDCG @10                 = 0.300377012553
Precision @10            = 0.247619047619
Reciprocal Rank          = 0.577097505669
R-Precision              = 0.267135073655
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester

```

Figure 9: R-precision and other evaluation measures for the entire CACM query set

```

1  #!/usr/bin/python
2
3  import errno
4  import os
5  from optparse import OptionParser
6  from lib.galago import GalagoRank
7
8  if __name__ == '__main__':
9      parser = OptionParser(description='Generate a ranking using Galago')
10     parser.set_usage(parser.get_usage().replace('\n', ' ') + ' <xml_file_input>')
11     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",
12                       help='Galago "home" directory')
13     parser.add_option('-d', '--document', dest="document_dir", default="/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm",
14                       help='Document directory to be indexed')
15     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",
16                       help='File .rel as galago eval judgments')
17     parser.add_option('-r', '--result', dest="result_count", default='10', help='Number of result')

```

```

18
19 (options, args) = parser.parse_args()
20 options = vars(options)
21
22 if len(args) < 1:
23     parser.print_help()
24     exit()
25
26
27 out_dir = os.path.abspath('output')
28
29 try: os.makedirs(out_dir)
30 except OSError, e:
31     if e.errno != errno.EEXIST: raise
32
33 index_dir = os.path.join(os.path.pardir, 'index')
34 xml_query_file = os.path.abspath(args[0])
35 json_query_file = os.path.join(out_dir, 'query.json')
36 rel_file = os.path.join(out_dir, 'result.rel')
37 res_file = os.path.join(out_dir, 'result.res')
38 eval_file = os.path.join(out_dir, 'result.eval')
39
40 galago = GalagoRank(options['galago_bin'], options['judgements_file'])
41 if galago.index(options['document_dir'], index_dir):
42     maps = []
43     ndcg_5s = []
44     ndcg_10s = []
45     prec_10s = []
46     recip_ranks = []
47     r_precs = []
48
49     for q_id in range(1, 64):
50         print 'Processing Query #{0}'.format(q_id)
51
52         json_query_file = os.path.join(out_dir, 'query_{0}.json'.format(q_id))
53         res_file = os.path.join(out_dir, 'result_{0}.res'.format(q_id))
54         eval_file = os.path.join(out_dir, 'result_{0}.eval'.format(q_id))
55
56         json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)
57
58         rel_docs = galago.get_relevance_docs(q_id)
59         res_docs = galago.search(index_dir, json_query_file, res_file, max(len(rel_docs), int
            (options['result_count'])))
60
61         galago.eval(options['judgements_file'], res_file, eval_file)
62
63         maps.append(galago.get_map(rel_docs, res_docs))
64         ndcg_5s.append(galago.get_ndcg(5, rel_docs, res_docs))
65         ndcg_10s.append(galago.get_ndcg(10, rel_docs, res_docs))
66         prec_10s.append(galago.get_all_precisions(rel_docs, res_docs)[9])
67         recip_ranks.append(galago.get_reciprocal_rank(rel_docs, res_docs))
68         r_precs.append(galago.get_r_precision(rel_docs, res_docs))
69
70
71 print 'Mean Average Precision    = {}'.format(float(sum(maps)) / len(maps))
72 print 'NDCG @5                  = {}'.format(float(sum(ndcg_5s)) / len(ndcg_5s))
73 print 'NDCG @10                 = {}'.format(float(sum(ndcg_10s)) / len(ndcg_10s))
74 print 'Precision @10            = {}'.format(float(sum(prec_10s)) / len(prec_10s))

```

```

75 print 'Reciprocal Rank          = {}'.format(float(sum( recip_ranks)) / len(
    recip_ranks))
76 print 'R-Precision              = {}'.format(float(sum(r_precs)) / len(r_precs))

```

Listing 5: Script for calculating R-precision

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:

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

and

$$BPREF = \frac{P}{P + Q} \quad (6)$$

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 10 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 10: BPREF values for query no. 3

From figure 10, 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 6

```

1  #!/usr/bin/python
2
3  import errno
4  import os
5  from optparse import OptionParser
6  from lib.galago import GalagoRank
7
8  if __name__ == '__main__':
9      parser = OptionParser(description='Generate a ranking using Galago')
10     parser.set_usage(parser.get_usage().replace('\n', ' ') + ' <xml_file_input> [q1 ... qn
        ]')
11     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',
12     help='Galago "home" directory')
13     parser.add_option('-d', '--document', dest="document_dir", default='/media/erikaris/
        DATA/ODU/Semester_3/intro_to_info_retrieval/assignments/a4/code-report/cacm',
14     help='Document directory to be indexed')
15     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',
16     help='File .rel as galago eval judgments')
17     parser.add_option('-r', '--result', dest="result_count", default='10', help='Number
        of result')
18
19     (options, args) = parser.parse_args()
20     options = vars(options)
21
22     if len(args) < 2:
23         parser.print_help()
24         exit()
25
26
27     out_dir = os.path.abspath('output')
28
29     try: os.makedirs(out_dir)
30     except OSError, e:
31         if e.errno != errno.EEXIST: raise
32
33     index_dir = os.path.join(os.path.pardir, 'index')
34     xml_query_file = os.path.abspath(args[0])
35     json_query_file = os.path.join(out_dir, 'query.json')
36     rel_file = os.path.join(out_dir, 'result.rel')
37     res_file = os.path.join(out_dir, 'result.res')
38     eval_file = os.path.join(out_dir, 'result.eval')
39
40     q_ids = []
41     if len(args) > 1:
42         q_ids = args[1:]
43
44     galago = GalagoRank(options['galago_bin'], options['judgements_file'])
45     if galago.index(options['document_dir'], index_dir):
46         for q_id in q_ids:
47             json_query = galago.build_json_input(xml_query_file, json_query_file, q_id)
48
49             rel_docs = galago.get_relevance_docs(q_id)
50             res_docs = galago.search(index_dir, json_query_file, res_file, options['result_count'
                ])
51
52     galago.eval(options['judgements_file'], res_file, eval_file)

```

```

53
54 print 'Query #'           = {}.format(json_query['number'])
55 print 'Query String'      = {}.format(json_query['text'])
56 print 'Relevant Documents' = {}.format(', '.join(rel_docs))
57 print 'Search Result Documents' = {}.format(', '.join(res_docs))
58 print 'BPREF-1'          = {}.format(galago.get_bpref_1(rel_docs, res_docs))
59 print 'BPREF-2'          = {}.format(galago.get_bpref_2(rel_docs, res_docs))

```

Listing 6: 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].