MSCI 541 – Search Engines

Homework 2

Professor Mark Smucker

Matthew Erxleben

ID: 20889980

Date: October 20th, 2023

Problem 2:

For program 1, I modified IndexEngine.py to tokenize all document terms, save those terms to a lexicon, and create an inverted index to track the frequency of term appearance in each document (if it contains the term). These files were saved to the disk, to be used for the BooleanAND.py program. In program 2, I created a program to perform Boolean AND Intersect retrieval. This program is called BooleanAND.py and is stored in the GitHub repository.

For each query, I turn the query terms into tokens (using the same methodology as I did in IndexEngine.py). I then load the inverted index, lexicon, and list of docno's into memory, to utilize them for fast retrieval.

For each query:

1. I tokenize the query terms and save them as tokens
2. I obtain the term_ids for each token buy using the lexicon
3. I obtain the postings list for each term using the inverted index
4. I then run the BooleanAND intersect program for each term's postings list, and find which documents contain all the query terms

After obtaining the BooleanAND result, I calculated the rank and score. I then output these results into a text file.

Some extra logic added to my BooleanAND program:

- If a query term is not present in ANY of the documents (therefore it does not exist in the lexicon), the query term is removed from the query completely. The query will then run with whatever terms it has left, or return nothing if there are no remaining query terms. This is because if this was not in place, the query would return nothing, even if the other query terms are a genuine search. The user may make a spelling error in their query, and they should not have a query return "nothing" because of this. This is due to BooleanAND being the retrieval method, which highlights its inefficiency as a retrieval algorithm. However, this has been put in place to make the method more functional.

To test BooleanAND's functionality, I created 5 documents. I utilized ChatGPT by giving it the latimes format I wanted the documents in and having it write the documents in the latimes format for me. Here is the text file that contains all 5 of the documents, in the latimes format:

```
<DOC>
<DOCNO>LA040490-0006</DOCNO>
<HEADLINE>
Breaking News: Earthquake in California
<P>
Residents Panic
</P>
</HEADLINE>
<TEXT>
A strong earthquake struck California today. Residents reported feeling the tremors.
<P>
The earthquake's epicenter was near Los Angeles.
</P>
</TEXT>
<GRAPHIC>
Image: People evacuating buildings during the earthquake.
</GRAPHIC>
</DOC>
<DOC>
<DOCNO>LA041390-0010</DOCNO>
<HEADLINE>
New Study on Climate Change!
</HEADLINE>
<TEXT>
A new study has been published on the effects of climate change.
It highlights the need for immediate action
<P>
The study's findings suggest rising global temperatures.
The NBA Finals have also been impacted
</P>
</TEXT>
<GRAPHIC>
Image: Melting ice caps in the Arctic.
<P>
Climate Crisis
</P>
</GRAPHIC>
</DOC>
<DOC>
<DOCNO>LA090490-0002</DOCNO>
<HEADLINE>
Sports Update: NBA Finals
<P>
Intense Showdown
</P>
</HEADLINE>
<TEXT>
The NBA Finals are in full swing, with the two top teams battling it out for the championship title.
</TEXT>
<GRAPHIC>
Image: Basketball players in action on the court.
</GRAPHIC>
</DOC>
<DOC>
```

```
<DOC>
<DOCNO>LA031789-0009</DOCNO>
<HEADLINE>
Tech News: Latest Smartphone Launch
</HEADLINE>
<TEXT>
The latest smartphone has been launched, featuring cutting-edge technology and a stunning design.
<P>
Tech enthusiasts are eager to get their hands on this new device.
Movie documentaries on these tech products will soon premiere with some hollywood stars in the cast
</P>
</TEXT>
<GRAPHIC>
Image: The new smartphone with its sleek design. Not very friendly towards climate change, as found in a recent study though.
</GRAPHIC>
</DOC>
<DOC>
<DOCNO>LA031789-0014</DOCNO>
<HEADLINE>
Entertainment: Movie Premiere
</HEADLINE>
<TEXT>
A highly anticipated movie had its premiere last night, with Hollywood stars in attendance.
</TEXT>
<GRAPHIC>
Image: Red carpet event at the
movie premiere.
</GRAPHIC>
</DOC>
```

I then gzipped this text file and then ran my IndexEngine.py file with this as data input. The IndexEngine.py file is then run and the files and meta data are saved, and the inverted index, doc lengths, and lexicon are created and saved to the disk as well:

doc_lengths.txt	2023-10-24 6:50 PM	Text Document
inverted_index.json	2023-10-24 6:50 PM	JSON File
lexicon.json	2023-10-24 6:50 PM	JSON File
lexicon_id_to_term.json	2023-10-24 6:50 PM	JSON File

The lexicon files can be seen here:

lexicon_id_to_term.json

File Edit View


```
{"0": "breaking", "1": "news", "2": "earthquake", "3": "in", "4": "california", "5": "residents", "6": "panic", "7": "a", "8": "strong", "9": "struck",
"10": "today", "11": "reported", "12": "feeling", "13": "the", "14": "tremors", "15": "s", "16": "epicenter", "17": "was", "18": "near", "19": "los",
"20": "angeles", "21": "image", "22": "people", "23": "evacuating", "24": "buildings", "25": "during", "26": "new", "27": "study", "28": "on", "29":
"climate", "30": "change", "31": "has", "32": "been", "33": "published", "34": "effects", "35": "of", "36": "it", "37": "highlights", "38": "need", "39":
"for", "40": "immediate", "41": "action", "42": "findings", "43": "suggest", "44": "rising", "45": "global", "46": "temperatures", "47": "nba", "48":
"finals", "49": "have", "50": "also", "51": "impacted", "52": "melting", "53": "ice", "54": "caps", "55": "arctic", "56": "crisis", "57": "sports", "58":
"update", "59": "intense", "60": "showdown", "61": "are", "62": "full", "63": "swing", "64": "with", "65": "two", "66": "top", "67": "teams", "68":
"battling", "69": "out", "70": "championship", "71": "title", "72": "basketball", "73": "players", "74": "court", "75": "tech", "76": "latest", "77":
"smartphone", "78": "launch", "79": "launched", "80": "featuring", "81": "cutting", "82": "edge", "83": "technology", "84": "and", "85": "stunning",
"86": "design", "87": "enthusiasts", "88": "eager", "89": "to", "90": "get", "91": "their", "92": "hands", "93": "this", "94": "device", "95": "movie",
"96": "documentaries", "97": "these", "98": "products", "99": "will", "100": "soon", "101": "premiere", "102": "some", "103": "hollywood", "104":
"stars", "105": "cast", "106": "its", "107": "sleek", "108": "not", "109": "very", "110": "friendly", "111": "towards", "112": "as", "113": "found",
"114": "recent", "115": "though", "116": "entertainment", "117": "highly", "118": "anticipated", "119": "had", "120": "last", "121": "night", "122":
"attendance", "123": "red", "124": "carpet", "125": "event", "126": "at"}
```


lexicon.json

File Edit View


```
{"breaking": 0, "news": 1, "earthquake": 2, "in": 3, "california": 4, "residents": 5, "panic": 6, "a": 7, "strong": 8, "struck": 9, "today": 10,
"reported": 11, "feeling": 12, "the": 13, "tremors": 14, "s": 15, "epicenter": 16, "was": 17, "near": 18, "los": 19, "angeles": 20, "image": 21,
"people": 22, "evacuating": 23, "buildings": 24, "during": 25, "new": 26, "study": 27, "on": 28, "climate": 29, "change": 30, "has": 31, "been": 32,
"published": 33, "effects": 34, "of": 35, "it": 36, "highlights": 37, "need": 38, "for": 39, "immediate": 40, "action": 41, "findings": 42, "suggest":
43, "rising": 44, "global": 45, "temperatures": 46, "nba": 47, "finals": 48, "have": 49, "also": 50, "impacted": 51, "melting": 52, "ice": 53, "caps":
54, "arctic": 55, "crisis": 56, "sports": 57, "update": 58, "intense": 59, "showdown": 60, "are": 61, "full": 62, "swing": 63, "with": 64, "two": 65,
"top": 66, "teams": 67, "battling": 68, "out": 69, "championship": 70, "title": 71, "basketball": 72, "players": 73, "court": 74, "tech": 75, "latest":
76, "smartphone": 77, "launch": 78, "launched": 79, "featuring": 80, "cutting": 81, "edge": 82, "technology": 83, "and": 84, "stunning": 85, "design":
86, "enthusiasts": 87, "eager": 88, "to": 89, "get": 90, "their": 91, "hands": 92, "this": 93, "device": 94, "movie": 95, "documentaries": 96, "these":
97, "products": 98, "will": 99, "soon": 100, "premiere": 101, "some": 102, "hollywood": 103, "stars": 104, "cast": 105, "its": 106, "sleek": 107, "not":
108, "very": 109, "friendly": 110, "towards": 111, "as": 112, "found": 113, "recent": 114, "though": 115, "entertainment": 116, "highly": 117,
"anticipated": 118, "had": 119, "last": 120, "night": 121, "attendance": 122, "red": 123, "carpet": 124, "event": 125, "at": 126}
```


Ln 1, Col 1718 | 100% | Windows (CRLF) | UTF-8

And the doc_lengths and inverted_index files can be seen here:

doc_lengths.txt

File Edit View

```
33
48
33
68
25
```

inverted_index.json

File Edit View


```
{"0": [0, 1], "1": [0, 1, 3, 1], "2": [0, 4], "3": [0, 1, 1, 1, 2, 2, 3, 2, 4, 1], "4": [0, 2], "5": [0, 2], "6": [0, 1], "7": [0, 1, 1, 1, 3, 2, 4, 1],
"8": [0, 1], "9": [0, 1], "10": [0, 1], "11": [0, 1], "12": [0, 1], "13": [0, 3, 1, 5, 2, 4, 3, 3, 4, 1], "14": [0, 1], "15": [0, 1, 1, 1], "16": [0, 1],
"17": [0, 1], "18": [0, 1], "19": [0, 1], "20": [0, 1], "21": [0, 1, 1, 1, 2, 1, 3, 1, 4, 1], "22": [0, 1], "23": [0, 1], "24": [0, 1], "25": [0, 1],
"26": [1, 2, 3, 2], "27": [1, 3, 3, 1], "28": [1, 2, 2, 1, 3, 2], "29": [1, 3, 3, 1], "30": [1, 2, 3, 1], "31": [1, 1, 3, 1], "32": [1, 2, 3, 1], "33":
[1, 1], "34": [1, 1], "35": [1, 1], "36": [1, 1, 2, 1], "37": [1, 1], "38": [1, 1], "39": [1, 1, 2, 1], "40": [1, 1], "41": [1, 1, 2, 1], "42": [1, 1],
"43": [1, 1], "44": [1, 1], "45": [1, 1], "46": [1, 1], "47": [1, 1, 2, 2], "48": [1, 1, 2, 2], "49": [1, 1], "50": [1, 1], "51": [1, 1], "52": [1, 1],
"53": [1, 1], "54": [1, 1], "55": [1, 1], "56": [1, 1], "57": [2, 1], "58": [2, 1], "59": [2, 1], "60": [2, 1], "61": [2, 1, 3, 1], "62": [2, 1], "63":
[2, 1], "64": [2, 1, 3, 2, 4, 1], "65": [2, 1], "66": [2, 1], "67": [2, 1], "68": [2, 1], "69": [2, 1], "70": [2, 1], "71": [2, 1], "72": [2, 1], "73":
[2, 1], "74": [2, 1], "75": [3, 3], "76": [3, 2], "77": [3, 3], "78": [3, 1], "79": [3, 1], "80": [3, 1], "81": [3, 1], "82": [3, 1], "83": [3, 1], "84":
[3, 1], "85": [3, 1], "86": [3, 2], "87": [3, 1], "88": [3, 1], "89": [3, 1], "90": [3, 1], "91": [3, 1], "92": [3, 1], "93": [3, 1], "94": [3, 1], "95":
[3, 1, 4, 3], "96": [3, 1], "97": [3, 1], "98": [3, 1], "99": [3, 1], "100": [3, 1], "101": [3, 1, 4, 3], "102": [3, 1], "103": [3, 1, 4, 1], "104": [3,
1, 4, 1], "105": [3, 1], "106": [3, 1, 4, 1], "107": [3, 1], "108": [3, 1], "109": [3, 1], "110": [3, 1], "111": [3, 1], "112": [3, 1], "113": [3, 1],
"114": [3, 1], "115": [3, 1], "116": [4, 1], "117": [4, 1], "118": [4, 1], "119": [4, 1], "120": [4, 1], "121": [4, 1], "122": [4, 1], "123": [4, 1],
"124": [4, 1], "125": [4, 1], "126": [4, 1]}
```

I then created 5 queries, with respective topic ids. These queries have expected documents that the BooleanAND intersect program should return.

Topic	Query	Expected Document to Return
101	California earthquake	LA040490-0006
102	climate change study	LA041390-0010, LA031789-0009
103	NBA finals	LA041390-0010, LA090490-0002
104	latest smartphone	LA031789-0009
105	movie premiere Hollywood stars	LA031789-0009, LA031789-0014

- Topic 101: LA040490-0006 contains both “california” and “earthquake” tokens, therefore it should be returned
- Topic 102: LA041390-0010, LA031789-0009 contain both “climate”, “change”, and “study” tokens, therefore it should be returned
- Topic 103: LA041390-0010, LA090490-0002 contain both “nba”, “finals” tokens, therefore it should be returned
- Topic 104: LA031789-0009 contains both “latest”, “smartphone” tokens, therefore it should be returned
- Topic 105: LA031789-0009, LA031789-0014 contains both “movie”, “premiere”, “hollywood”, “stars” tokens, therefore it should be returned

I created a queries.txt file that contains the queries in the format topic number, (newline), and then the query. This can be seen here:

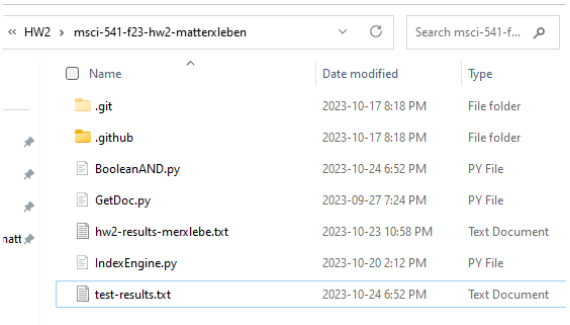
```
101
California earthquake
102
climate change study
103
NBA finals
104
latest smartphone
105
movie premiere Hollywood stars
```

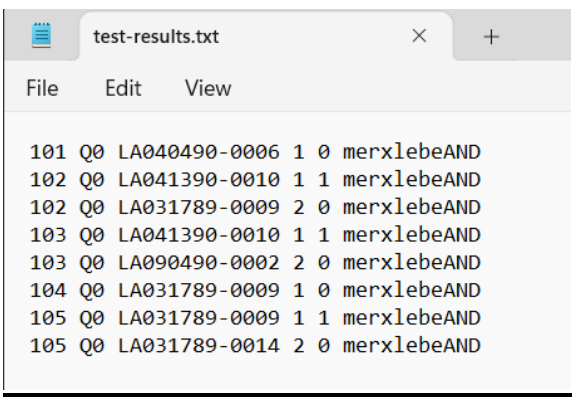
When running the BooleanAND program from the terminal with the following input:

python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/test-docs queries.txt test-results.txt

The queries are tokenized, and the lexicon and inverted index are used to obtain the postings list for each term in the given query. The BooleanAND intersect algorithm is run on these postings lists to obtain documents where the query terms are present in all documents for a given query. These results are then saved in the test-results.txt file.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben> python Boolean
AND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/test-docs queries.txt test-results.txt
Done loading inverted index!
['california', 'earthquake']
['climate', 'change', 'study']
['nba', 'finals']
['latest', 'smartphone']
['movie', 'premiere', 'hollywood', 'stars']
Output saved!
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben>
```





As we can see, the results of the BooleanAND intersect retrieval returned the same documents as the expected documents outlined previously. This demonstrates that BooleanAND retrieval is fully functional in this program and obtains the correct results. The results are also in the format requested in the HW2 requirements: topicID Q0 docno rank score runTag.

Installation Requirements:

1. Please make sure Python is installed on your computer before running the program.

2. Clone repository on your device by entering this into your terminal: git clone https://github.com/UWaterloo-MSCI-541/msci-541-f23-hw2-matterxleben.git

Running the Programs:

In order to run these programs, please navigate to where you cloned the repository and open the working directory .../msci-541-f23-hw2-matterxleben

IndexEngine:

This program accepts two command line arguments:

1. a path to the latimes.gz file
2. a path to a directory where the documents, metadata, and term files will be stored.

For example, you would run IndexEngine from the command prompt / terminal / shell as:

python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/store

GetDoc:

The program accepts three command line arguments:

1. a path to the location of the documents and metadata store created by the first program (IndexEngine)
2. either the string "id" or the string "docno"
3. either the internal integer id of a document or a DOCNO

For example, you would run GetDoc from the command prompt / terminal / shell as:

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/store docno LA010189-0003

OR

python GetDoc.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/store id 2

BooleanAND:

The program accepts three command line arguments: the directory location of your index, the queries file, and the name of a file to store your output

1. a path to the location of your index, created by IndexEngine
2. the queries file
3. the name of a file to store your output

For example, you would run BooleanAND from the command prompt / terminal / shell as:

python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/store/term_files queries.txt test-results.txt

For this program, the queries file (search topics file) is in the /topics files folder. In the code for the program, the path is hardcoded as:

C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/msci-541-f23-hw2-matterxleben/topics files/" + queries_file

Please change this to your own computers path to the msci-541-f23-hw2-matterxleben/topics files/ folder to obtain the queries (search topics) !

Edge Cases:

1. **No arguments supplied to program:**

Prints out the response:

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben> python BooleanAND.py

            This input does not meet the requirements for this program!
            The BooleanAND program's goal is to efficiently retrieve documents and based on input queries from the user.

            The program accepts three command line arguments:
            the directory location of your index, the queries file, and the name of a file to store your output
                1. a path to the location of your index, created by IndexEngine
                2. the queries file
                3. the name of a file to store your output

            For example, you would run BooleanAND from the command prompt / terminal / shell as:
                python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/store/term_files queries.txt hw2-results-merxlebe.txt

PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben>
```

and exits the program

2. **More than 3 arguments supplied to program:**

Prints out the response:

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben> python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Sear
ch-Engines/HW2/test-docs queries.txt test-results.txt adadsda sdds

        This input does not meet the requirements for this program!
        The BooleanAND program's goal is to efficiently retrieve documents and based on input queries from the user.

        The program accepts three command line arguments:
        the directory location of your index, the queries file, and the name of a file to store your output
            1. a path to the location of your index, created by IndexEngine
            2. the queries file
            3. the name of a file to store your output

        For example, you would run BooleanAND from the command prompt / terminal / shell as:
            python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW2/store/term_files queries.txt hw2-results-merxlebe.txt
```

and exits the program

3. **Inverted Index path does not exist or is incorrect:**

Prints out the response: "This path does not exist! Please enter the index!", and exits the program

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben> python BooleanAND.py adadsadsffdfdfs queries.txt test-results.txt
This path does not exist! Please enter the correct path to the index!
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben>
```

4. **Queries text file does not exist or is incorrect:**

Prints out the response: "This path does not exist! Please enter the file name to your queriues!", and exits the program.

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben> python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Sea
ch-Engines/HW2/testasdasdsadsads queradasdsdsdadasdssdies.txt test-results.txt
This path does not exist! Please enter the correct file name for your queries!
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben>
```

5. **Output file already exists:**

Prints out the response: “The output file already exists! Please enter a new name for your results file, or delete the previously stored file and rerun this program!” and exists the program

```
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben> python BooleanAND.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Sea
ch-Engines/HW2/store/term_files queries.txt test-results.txt
Done loading inverted index!
The output file already exists! Please enter a new name for your results file, or delete the previously stored file and rerun this program!
PS C:\Users\matth\OneDrive\Desktop\University\3B\MSCI541-Search-Engines\HW2\msci-541-f23-hw2-matterxleben>
```

B) LATIMES and all 45 Queries:

This program was also run on the entire latimes dataset and using all 45 queries. The resulting file was stored to a file named hw2-results-merxlebe.txt, stored in the root of the GitHub repository. The file contains 2336 results (rows) for this data and these queries. Screen shots of the results can be seen below:

hw2-results-merxlebe.txt

```
401 Q0 LA021890-0100 1 13 merxlebeAND
401 Q0 LA040389-0047 2 12 merxlebeAND
401 Q0 LA040490-0003 3 11 merxlebeAND
401 Q0 LA050590-0114 4 10 merxlebeAND
401 Q0 LA050789-0068 5 9 merxlebeAND
401 Q0 LA051390-0170 6 8 merxlebeAND
401 Q0 LA052190-0065 7 7 merxlebeAND
401 Q0 LA082690-0052 8 6 merxlebeAND
401 Q0 LA090490-0093 9 5 merxlebeAND
401 Q0 LA100889-0019 10 4 merxlebeAND
401 Q0 LA111289-0073 11 3 merxlebeAND
401 Q0 LA121890-0117 12 2 merxlebeAND
401 Q0 LA122389-0060 13 1 merxlebeAND
401 Q0 LA122990-0070 14 0 merxlebeAND
402 Q0 LA030690-0001 1 2 merxlebeAND
402 Q0 LA062590-0042 2 1 merxlebeAND
402 Q0 LA101290-0115 3 0 merxlebeAND
403 Q0 LA010390-0067 1 43 merxlebeAND
403 Q0 LA010490-0218 2 42 merxlebeAND
403 Q0 LA010689-0040 3 41 merxlebeAND
403 Q0 LA010790-0103 4 40 merxlebeAND
403 Q0 LA011289-0149 5 39 merxlebeAND
403 Q0 LA011389-0029 6 38 merxlebeAND
403 Q0 LA012990-0041 7 37 merxlebeAND
403 Q0 LA020490-0136 8 36 merxlebeAND
403 Q0 LA020990-0100 9 35 merxlebeAND
403 Q0 LA021590-0062 10 34 merxlebeAND
403 Q0 LA022790-0099 11 33 merxlebeAND
403 Q0 LA030689-0082 12 32 merxlebeAND
403 Q0 LA032290-0151 13 31 merxlebeAND
403 Q0 LA032489-0093 14 30 merxlebeAND
403 Q0 LA033089-0013 15 29 merxlebeAND
403 Q0 LA033089-0019 16 28 merxlebeAND
403 Q0 LA041990-0072 17 27 merxlebeAND
```

hw2-results-merxlebe.txt

```
450 Q0 LA081890-0073 26 31 merxlebeAND
450 Q0 LA081890-0087 27 30 merxlebeAND
450 Q0 LA081990-0043 28 29 merxlebeAND
450 Q0 LA081990-0065 29 28 merxlebeAND
450 Q0 LA082190-0087 30 27 merxlebeAND
450 Q0 LA082690-0110 31 26 merxlebeAND
450 Q0 LA082790-0013 32 25 merxlebeAND
450 Q0 LA090290-0167 33 24 merxlebeAND
450 Q0 LA090290-0168 34 23 merxlebeAND
450 Q0 LA090990-0032 35 22 merxlebeAND
450 Q0 LA091290-0066 36 21 merxlebeAND
450 Q0 LA092290-0079 37 20 merxlebeAND
450 Q0 LA092390-0215 38 19 merxlebeAND
450 Q0 LA093090-0075 39 18 merxlebeAND
450 Q0 LA093090-0186 40 17 merxlebeAND
450 Q0 LA100189-0187 41 16 merxlebeAND
450 Q0 LA100190-0080 42 15 merxlebeAND
450 Q0 LA100490-0080 43 14 merxlebeAND
450 Q0 LA100590-0052 44 13 merxlebeAND
450 Q0 LA101290-0010 45 12 merxlebeAND
450 Q0 LA102890-0170 46 11 merxlebeAND
450 Q0 LA111490-0061 47 10 merxlebeAND
450 Q0 LA111890-0012 48 9 merxlebeAND
450 Q0 LA120490-0103 49 8 merxlebeAND
450 Q0 LA120790-0182 50 7 merxlebeAND
450 Q0 LA121090-0008 51 6 merxlebeAND
450 Q0 LA121090-0081 52 5 merxlebeAND
450 Q0 LA121390-0114 53 4 merxlebeAND
450 Q0 LA121589-0028 54 3 merxlebeAND
450 Q0 LA121690-0162 55 2 merxlebeAND
450 Q0 LA122690-0031 56 1 merxlebeAND
450 Q0 LA123090-0176 57 0 merxlebeAND
```

Problem 3:

Topic 401: "foreign minorities, Germany"

Rank	DOCNO	Relevance	Relevance Description
1	LA021890-0100	Not Relevant	"foreign" only gets mentioned once, while discussing a German minister. "minorities" only gets mentioned when discussing other countries groups of adults reluctant to see Germany's reconstruction. The article is more about Germany and political dangers, not the topic. This is not relevant
2	LA040389-0047	Not Relevant	This article is about NATO and its strategies regarding a new soviet president. This article is not discussing foreign minorities in Germany.
3	LA040490-0003	Not Relevant	This article discusses the Soviet legislature and the laws that are following the Soviet Unions split. This

			article is not discussing foreign minorities in Germany.
4	LA050590-0114	Not Relevant	Latvia declaration of independence. Does discuss ethnic minorities, not German or in Germany however.
5	LA050789-0068	Relevant	This article discusses a pros and cons of immigration in Europe, and does mention foreign minorities in Germany. These minorities can face limitations due to not being from the country. This can cause foreign minorities immigrating to Germany to face challenges in society. This is relevant.
6	LA051390-0170	Not Relevant	Baltic States and their relationship with the Soviet Union. Not discussing the topic, therefore not relevant.
7	LA052190-0065	Not Relevant	Romanian elections, and politics in Romania during the time of their election for the national salvation front. Not relevant.
8	LA082690-0052	Not Relevant	Discusses motorcyclists driving across the Silk Road, through the soviet union. Does mention minorities along the way from other countries, however not directly German or in Germany.
9	LA090490-0093	Not Relevant	The gulf crisis and how it affects Europeans. How it impacts policy and society in Europe. Not relevant.
10	LA100889-0019	Not Relevant	Congress discussing the limits and effects on writers that can come with the position. Some writers have been imprisoned in countries, such as eastern Europe. Not about the topic

Precision = $\frac{1}{10} = 10\%$

As we can see, only 1 of the documents I found to be relevant. Therefore, the precision is 10%.

Topic 403: "osteoporosis"

Rank	DOCNO	Relevance	Relevance Description
1	LA010390-0067	Relevant	Directly discusses osteoporosis, directly about its affects on men's health
2	LA010490-0218	Not Relevant	Discusses more general pharmaceutical breakthroughs, not directly osteoporosis

3	LA010689-0040	Not Relevant	Discussing wedding gowns, and bride and groom etiquette at weddings. Mentions osteoporosis in a quote, doesn't directly discuss it.
4	LA010790-0103	Relevant	Discussing medical research on osteoporosis and how to prevent the issue, specifically in women.
5	LA011289-0149	Not Relevant	Nutrition, weight loss, and dieting. The article breaks down recipes. Not about osteoporosis
6	LA011389-0029	Relevant	Discussed osteoporosis treatments, such as sodium and calcium supplements. Relevant
7	LA012990-0041	Not Relevant	Dairy Milk marking efforts. Not about osteoporosis
8	LA020490-0136	Relevant	Research into preventing osteoporosis. Relevant
9	LA020990-0100	Not Relevant	Discussing food regulations from the FDA, and stopping some bad food marketing. Not relevant
10	LA021590-0062	Not Relevant	Discussing nutrition and dieting to maintain healthy cholesterol levels. Mentions preventing osteoporosis but only once, not the purpose of the article and not relevant.

Precision = $\frac{4}{10} = 40\%$

As we can see, 4 of the documents I found to be relevant. Therefore, the precision is 40%.

In conclusion, the documents returned for topic 403 were much more often relevant than topic 401. This is likely due to topic 403 only having 1 query term, and it being an uncommon word in the average article. However, the precision is still low for both queries. This is due to BooleanAND being a poor retrieval method.