

Assignment 1 – CSI 4107

Group 4

Maria Stancu (300243486)

Hasan Hashim (300194035)

Trinity Vermeire (300129927)

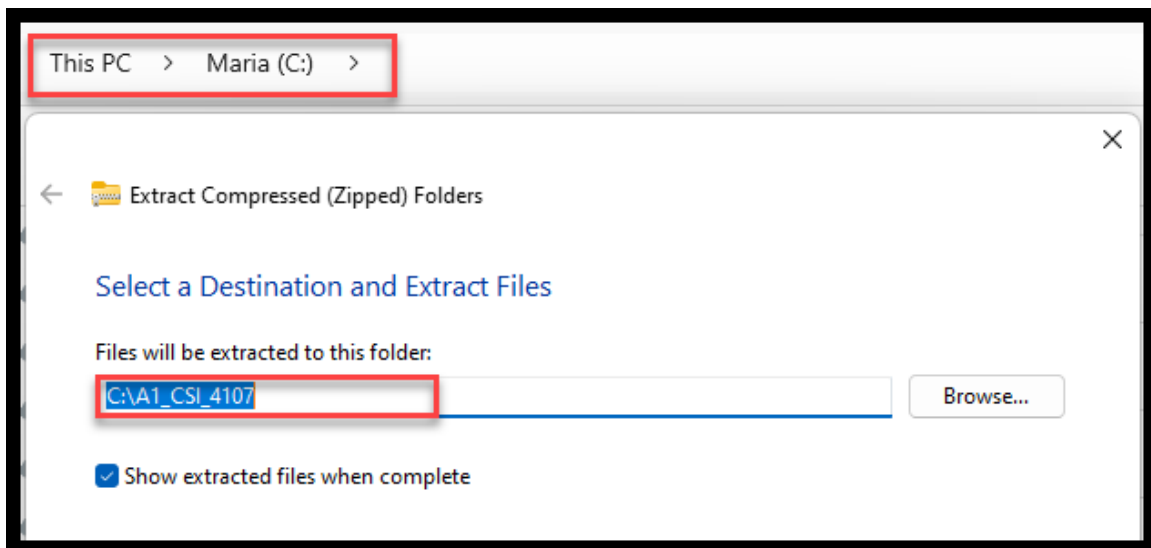
Contents

Environment	2
Results file	5
Mean Average Precision (MAP)	5
Java class files (compilation).....	6
Program execution.....	7
Search by similarity BM25 and by QTITLE	8
Search by similarity BM25 and by QDESC	8
Search by similarity tf-idf and by QTITLE	8
Search by similarity tf-idf and by QDESC	8
Introduction.....	9
Step 1 – Preprocessing	9
Step 2 – Indexing	9
Step 3 – Retrieval and ranking.....	10
Vocabulary size: 224224	10
100 tokens from the vocabulary (sample).....	11
Similarity.....	14
Include the first 10 answers to queries 1 and 25. Discuss your results.....	16
Mean Average Precision (MAP)	21
tf-idf / query title	21

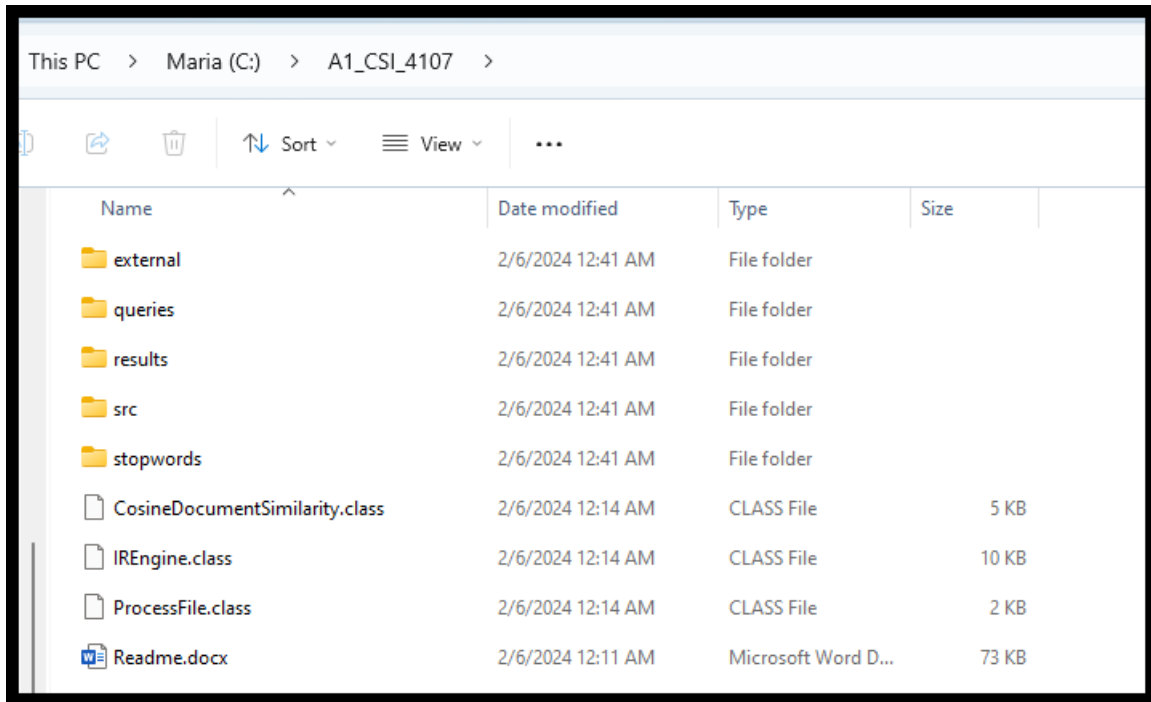
tf-idf / query description	22
bm25 / query title (best score).....	23
bm25 / query description	24
Conclusion	25
Task Distribution	25

Environment

- Copy the file A1_CSI_4107.zip to a your computer
- Unzip the file:



- It will create the following structure:



- The initial text collection is not included in the zip files.
 - To be able to compile the Java source files you need to create a folder named coll → place the collection files under the coll folder.

This PC > Maria (C:) > A1_CSI_4107 >

Sort View ...

Name	Date modified	Type	Size
coll	2/6/2024 12:50 AM	File folder	
external	2/6/2024 12:41 AM	File folder	
queries	2/6/2024 12:41 AM	File folder	
results	2/6/2024 12:41 AM	File folder	
src	2/6/2024 12:41 AM	File folder	
stopwords	2/6/2024 12:41 AM	File folder	
CosineDocumentSimilarity.class	2/6/2024 12:14 AM	CLASS File	5 KB
IREngine.class	2/6/2024 12:14 AM	CLASS File	10 KB
ProcessFile.class	2/6/2024 12:14 AM	CLASS File	2 KB
Readme.docx	2/6/2024 12:11 AM	Microsoft Word D...	73 KB

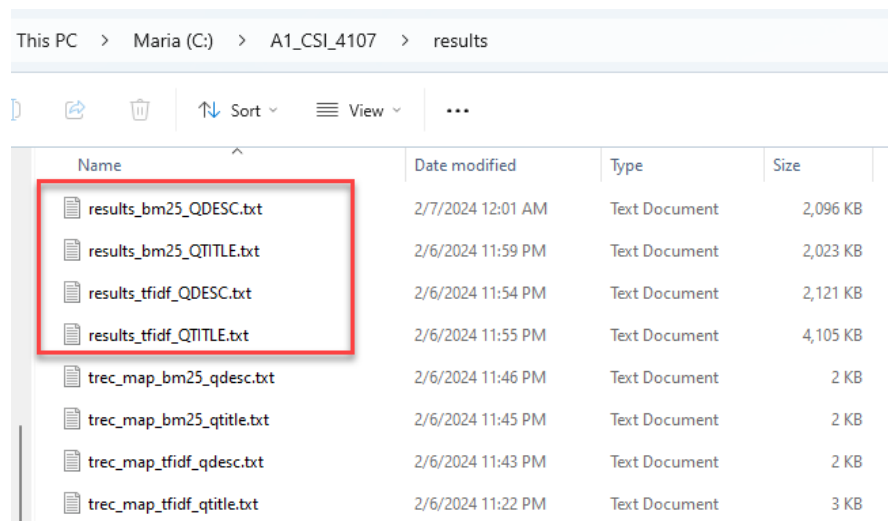
This PC > Maria (C:) > A1_CSI_4107 > coll

Sort View ...

Name	Date modified	Type	Size
AP880212	1/26/2024 10:46 AM	File	530 KB
AP880213	1/26/2024 10:46 AM	File	633 KB
AP880214	1/26/2024 10:46 AM	File	281 KB
AP880215	1/26/2024 10:46 AM	File	832 KB
AP880216	1/26/2024 10:46 AM	File	691 KB
AP880217	1/26/2024 10:46 AM	File	947 KB
AP880218	1/26/2024 10:46 AM	File	927 KB

Results file

- The results file is located under: **C:\A1_CSI_4107\results**
- There are 4 results files:
 - results_bm25_QDESC.txt – results created using BM25 similarity and search by query description.
 - results_bm25_QTITLE.txt – results created using the BM25 similarity and search by query title.
 - results_tfidf_QDESC.txt – results created using the Lucene tf-idf similarity and search by query description.
 - results_tfidf_QTITLE.txt – results created using Lucene tf-idf similarity and search by query title.



Name	Date modified	Type	Size
results_bm25_QDESC.txt	2/7/2024 12:01 AM	Text Document	2,096 KB
results_bm25_QTITLE.txt	2/6/2024 11:59 PM	Text Document	2,023 KB
results_tfidf_QDESC.txt	2/6/2024 11:54 PM	Text Document	2,121 KB
results_tfidf_QTITLE.txt	2/6/2024 11:55 PM	Text Document	4,105 KB
trec_map_bm25_qdesc.txt	2/6/2024 11:46 PM	Text Document	2 KB
trec_map_bm25_qtitle.txt	2/6/2024 11:45 PM	Text Document	2 KB
trec_map_tfidf_qdesc.txt	2/6/2024 11:43 PM	Text Document	2 KB
trec_map_tfidf_qtitle.txt	2/6/2024 11:22 PM	Text Document	3 KB

Mean Average Precision (MAP)

- The MAP score computed with trec_eval script is included under the folder: **C:\A1_CSI_4107\results**
- There is one MAP file for each corresponding results file

This PC > Maria (C:) > A1_CSI_4107 > results

Name	Date modified	Type	Size
results_bm25_QDESC.txt	2/7/2024 12:01 AM	Text Document	2,096 KB
results_bm25_QTITLE.txt	2/6/2024 11:59 PM	Text Document	2,023 KB
results_tfidf_QDESC.txt	2/6/2024 11:54 PM	Text Document	2,121 KB
results_tfidf_QTITLE.txt	2/6/2024 11:55 PM	Text Document	4,105 KB
trec_map_bm25_qdesc.txt	2/6/2024 11:46 PM	Text Document	2 KB
trec_map_bm25_qtitle.txt	2/6/2024 11:45 PM	Text Document	2 KB
trec_map_tfidf_qdesc.txt	2/6/2024 11:43 PM	Text Document	2 KB
trec_map_tfidf_qtitle.txt	2/6/2024 11:22 PM	Text Document	3 KB

Java class files (compilation)

- The Java files are already compiled, there is no need to recompile them.
- The Java class files are placed under the folder: C:\A1_CSI_4107

This PC > Maria (C:) > A1_CSI_4107 >

Name	Date modified	Type	Size
coll	2/6/2024 12:50 AM	File folder	
external	2/6/2024 12:41 AM	File folder	
queries	2/6/2024 12:58 AM	File folder	
results	2/6/2024 12:41 AM	File folder	
src	2/6/2024 12:41 AM	File folder	
stopwords	2/6/2024 12:41 AM	File folder	
CosineDocumentSimilarity.class	2/6/2024 12:14 AM	CLASS File	5 KB
IREngine.class	2/6/2024 12:14 AM	CLASS File	10 KB
ProcessFile.class	2/6/2024 12:14 AM	CLASS File	2 KB
Readme.docx	2/6/2024 12:11 AM	Microsoft Word D...	73 KB

- In case you want to compile again the source files, from the src folder execute the following command:

```
C:\Users\Maria24>cd C:\A1_CSI_4107\src

C:\A1_CSI_4107\src>dir
Volume in drive C is Maria
Volume Serial Number is 8E3D-0544

Directory of C:\A1_CSI_4107\src

02/06/2024  12:41 AM    <DIR>          .
02/06/2024  12:47 AM    <DIR>          ..
02/03/2024  05:34 PM                3,931 CosineDocumentSimilarity.java
02/06/2024  12:10 AM                17,713 IREngine.java
02/02/2024  12:06 PM                3,159 ProcessFile.java
               3 File(s)                24,803 bytes
               2 Dir(s)  490,220,593,152 bytes free
```

C:\A1_CSI_4107\src>javac -d ../ -cp ../external/*;*.java

The above command will update the .class files located under C:\A1_CSI_4107

Program execution

- To execute the program execute the following command from C:\A1_CSI_4107
 - **Prerequisite:** the collection of documents is available under the coll folder
- **Program has two parameters:** first is the SIMILARITY → choose between **tf-idf** or **BM25** and the next one is the query search parameter → choose between **QDESC** (= search by query description) or **QTITLE** (= search by query title).
- **The results files will be available under the C:\A1_CSI_4107\results folder**
- The execution will execute the following tasks:
 - Display the total number of words/tokens (including duplicates and special characters): 35,805,165
 - **Display the vocabulary size: 224224** = size of dictionary after the preprocessing using the stopwords (located under the /stopwords folder) and eliminating the duplicates
 - Displays the number of queries: 50
 - Update the results file(s) under the ./results folder:

Group 4
11/02/2024

Search by similarity BM25 and by QTITLE

From C:\A1_CSI_4107

```
java -cp ./external/*;. IREngine BM25 QTITLE
```

```
C:\A1_CSI_4107>java -cp ./external/*;. IREngine BM25 QTITLE
Total number of words: 35805165
Vocabulary size: 224224
Number of queries: 50
```

Search by similarity BM25 and by QDESC

From C:\A1_CSI_4107

```
java -cp ./external/*;. IREngine BM25 QDESC
```

```
C:\A1_CSI_4107>java -cp ./external/*;. IREngine BM25 QDESC
Total number of words: 35805165
Vocabulary size: 224224
Number of queries: 50
```

Search by similarity tf-idf and by QTITLE

From C:\A1_CSI_4107

```
java -cp ./external/*;. IREngine tf-idf QTITLE
```

```
C:\A1_CSI_4107>java -cp ./external/*;. IREngine tf-idf QTITLE
Total number of words: 35805165
Vocabulary size: 224224
Number of queries: 50
```

Search by similarity tf-idf and by QDESC

From C:\A1_CSI_4107

```
java -cp ./external/*;. IREngine tf-idf QDESC
```

```
C:\A1_CSI_4107>java -cp ./external/*;. IREngine tfidf QDESC
Total number of words: 35805165
Vocabulary size: 224224
Number of queries: 50
```


Introduction

For the implementation of the Information Retrieval (IR) system based on the vector space model, we chose Lucene. Apache Lucene is an open source project and it offers the possibility to implement a detailed full-featured text search.

The IR system was implemented in Java.

Step 1 – Preprocessing

We read the files which form the document corpus one by one. Each file contains a number of documents separated by specific tags: we are reading each file and each document from each file.

- First, we are saving the content of the following tags: DOCNO, FILEID, HEAD, DATELINE

```
for (int i=0;i<docs.length;i++) {  
    // For each document within the file, get the document number  
    DOCNO = fContent.getDOCNO(docs[i]);  
    FILEID = fContent.getFileID(docs[i]);  
    HEAD = fContent.getHEAD(docs[i]);  
    DATELINE = fContent.getDateLine(docs[i]);  
}
```

- Then, we are removing the tags from each document – the tags are not part of the vocabulary.

For removing the tags, we are using a regex expression:

```
doc =  
oneDoc.replaceAll("(?m)^(<DOC>|<DOCNO>.*|<FILEID>.*|<HEAD>.*|<1ST_LINE>.*|<2ND_LINE>.*)(?:\\r?\\n)?", "");
```

After removing the tags, we split by space: in this way we are creating the tokens.

- At the same time, we are counting the total number of tokens (words): 35,805,165
 - This is not the vocabulary size; this represents the total number of tokens including duplicates.
- We initialize the stopwords by using the stopwords file provided in the project: the stopwords will eliminate the words which are not relevant for searches.

Step 2 – Indexing

Lucene index writer is based on the concept of the Analyzer: the analyzer is initiated using the stopwords and the inverted index is created having the Analyzer as an argument: the stopwords are eliminated before indexing the words. For this project, the index is saved in the memory –

but the index could be saved as a file at the operating system level or even inside a RDBMS (i.e. Access)

```
// *****  
// *          SETUP THE ANALYZER          *  
// *****  
stopWords = getStopwords();  
CharArrayList stopSet = new CharArrayList(stopWords, true);  
analyzer = new StandardAnalyzer(stopSet);
```

The inverted index is updated with the content of each document.

We are storing the DOCNO, FILEID, HEAD, DATELINE as stored fields: we can access these fields when we are retrieving the document through a query search.

We are storing the content of the document under the label “text” in the inverted index.

```
public static void buildIndex(String line, String docno, String fileid, String head,  
    Document docu = new Document();  
    docu.add(new StringField("docno", docno, Field.Store.YES));  
    docu.add(new StringField("fileid", docno, Field.Store.YES));  
    docu.add(new StringField("head", docno, Field.Store.YES));  
    docu.add(new StringField("dateline", docno, Field.Store.YES));  
    docu.add(new TextField("text", line, Field.Store.NO));  
    w.addDocument(docu);  
}
```

Step 3 – Retrieval and ranking

At this time, the inverted index has been created and words have been indexed for each document. We preserved the DOCNO, FILEID, HEAD, DATELINE for each document while the document content is labeled “text”

For the retrieval and ranking, we are opening the index for reading. At this time, we are also accessing all the words available in the index (the vocabulary). Using an iterator we are looping through the index and get the vocabulary size.

Vocabulary size: 224224

```
// *****  
// *          VOCABULARY SIZE          *  
// *****  
LuceneDictionary ld = new LuceneDictionary( reader, "text" );  
// BytesRefIterator iterator = ld.getgetWordsIterator();  
InputIterator iterator = ld.getEntryIterator();
```

We are only counting the words under the label “text” in the index.

```
C:\A1_CSI_4107>java -cp ./external/*;. IREngine BM25 QDESC
Total number of words: 35805165
Vocabulary size: 224224
Number of queries: 50
```

The vocabulary size is: 224224 words – this total number of words includes also existing non-duplicated numbers from our document corpus content.

100 tokens from the vocabulary (sample)

munfordville

mung

mungai

mungar

mungari

mungbeans

munger

mungo

munhall

munhango

muni

munib

munich

munichois

municipal

municipales

municipalities

municipality

municipally

municipals

municipio

munier

muniinsured

Group 4
11/02/2024

munijcipal

munim

municipalities

munir

munira

munis

muniserable

munist

munition

munitions

muniyandi

muniz

munjed

munk

munkacsy

munley

munn

munnetra

munninghoff

munno

munns

munos

munoz

munro

munroe

munsan

munsch

munsel

munsell

Group 4
11/02/2024

munshi

munshiganj

munsinger

munsingwear

munson

munster

munstergeleen

munsterman

munsters

munt

muntasser

muntjac

muntoni

munusamy

munyakayanza

munyarara

muoi

muon

muons

muotathal

muphy

muppet

muppeteer

muppets

muqassed

muqattam

muqtadir

mura

murabito

Group 4
11/02/2024

murad
muradyan
murakami
murakawa
murakhovsky
mural
murali
muralist
muralists
murals
muramatso
muramatsu
muramvya
muranaka
murano
muranski
muranyi
muraoka
murari

Similarity

In Lucene we have the option of selecting the desired similarity used for ranking documents.

By using parameters when calling the program: (i.e. `java -cp ./external/*;. IREngine tf-idf QTITLE`) we could toggle the similarity between tf-idf or BM25. The BM25 similarity is the default for Lucene. The parameters names related to similarity are **tf-idf** or **BM25**

Lucene does not use anymore (in the latest release) the cosine similarity because the tf-idf and the BM25 similarities produce better results.

We created a Java class: CosineDocumentSimilarity which calculates the cosine similarity between two vectors: the query and the document vector. But our further calculations are based on the tf-idf or BM25 similarity.

To calculate the ranking and execute the retrieval we prepared the queries:

- Read the file containing all the queries

```
String[] queries = prepareQueries("./queries/50queries.txt");  
//String[] queries = prepareQueries("./queries/qury_test.txt");  
System.out.println("Number of queries: " + (queries.length-1));
```

- Identified each query by splitting the content of the file on the </top> tag.
- To allow searches by title or by description or by narrative, we save the following values: query number, description, title, and narrative.

```
QNUM = getQNUM(queries[a]);  
QDESC = getQDESC(queries[a]);  
QTITLE = getQTITLE(queries[a]);  
QNARR = getQNARR(queries[a]);
```

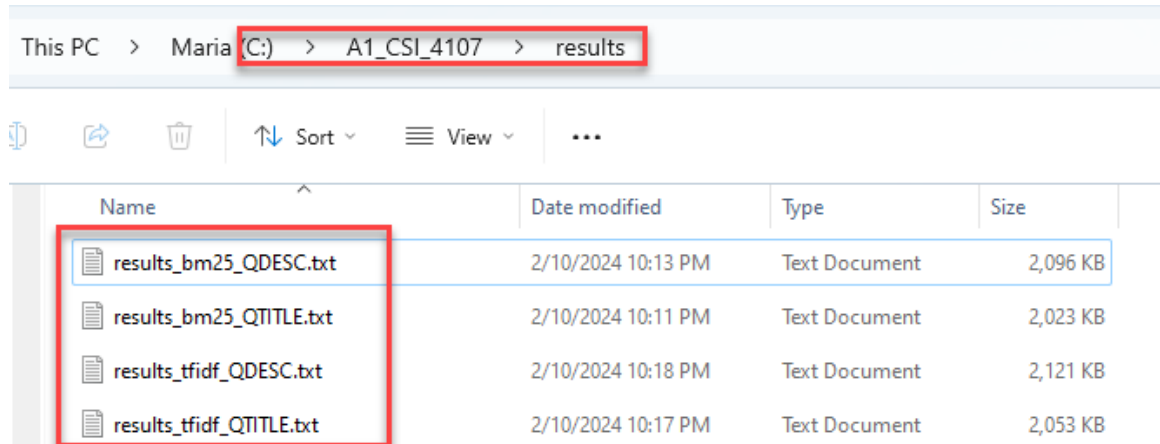
- We set the number of hits per page to 1000.

```
int hitsPerPage = 1000;
```

- At this point, we have chosen the similarity (tf-idf or BM25)
- Based on parameters passed, the search can be either by query title or by query description: parameters QTITLE or QDESC

```
String querystr;  
if (querySearch.equals("QTITLE")) {  
    querystr = QTITLE;  
} else if (querySearch.equals("QDESC")) {  
    querystr = QDESC;  
} else {  
    querystr = QTITLE;  
}
```

- We are executing the retrieval and store the number of hitsPerPage = 1000
- For each combination chosen we are creating a different file under the results folder



- The content of each file corresponds to the required format:

results_bm25_QDESC - Notepad

File Edit Format View Help

```
1 Q0 AP880815-0061 1 18.87426 STANDARD
1 Q0 AP881206-0124 2 18.584637 STANDARD
1 Q0 AP881108-0076 3 17.540321 STANDARD
1 Q0 AP881005-0001 4 16.57506 STANDARD
1 Q0 AP880814-0089 5 16.140993 STANDARD
1 Q0 AP880926-0180 6 15.910308 STANDARD
1 Q0 AP881223-0053 7 15.846453 STANDARD
1 Q0 AP880825-0054 8 15.796862 STANDARD
1 Q0 AP881202-0169 9 15.1591 STANDARD
1 Q0 AP880726-0173 10 14.066102 STANDARD
1 Q0 AP881002-0014 11 13.74097 STANDARD
1 Q0 AP881225-0044 12 12.947003 STANDARD
1 Q0 AP881211-0022 13 12.91345 STANDARD
1 Q0 AP880505-0074 14 12.218069 STANDARD
1 Q0 AP881122-0004 15 12.153965 STANDARD
1 Q0 AP880324-0249 16 12.134049 STANDARD
1 Q0 AP880607-0210 17 12.066861 STANDARD
```

Include the first 10 answers to queries 1 and 25. Discuss your results.

- The similarity used in the following example is tf-idf
- When Lucene does its scoring, it sums up a set of scores together to give a total score.

The Query #1 has the following characteristics:

<top>

<num>1

<title>Coping with overcrowded prisons

<desc>

The document will provide information on jail and prison overcrowding and

how inmates are forced to cope with those conditions; or it will reveal plans to relieve the overcrowded condition.

<narr>

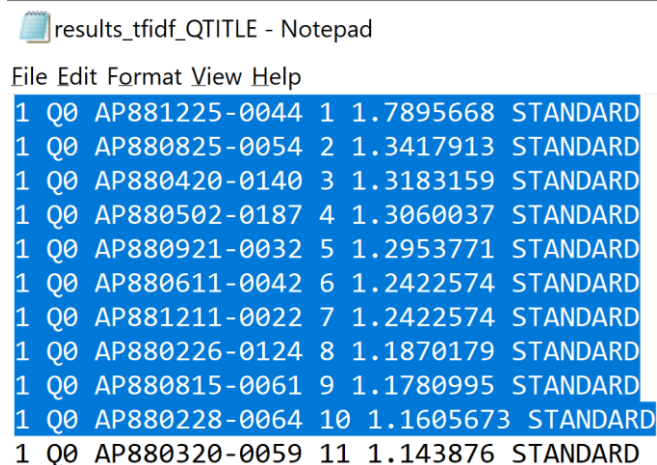
A relevant document will describe scenes of overcrowding that have become all too common in jails and prisons around the country. The document will identify how inmates are forced to cope with those overcrowded conditions, and/or what the Correctional System is doing, or planning to do, to alleviate the crowded condition.

</top>

➔ The retrieval was performed using the query title:

Coping with overcrowded prisons

⇒ It produced the following result (top 10)



```
File Edit Format View Help
1 Q0 AP881225-0044 1 1.7895668 STANDARD
1 Q0 AP880825-0054 2 1.3417913 STANDARD
1 Q0 AP880420-0140 3 1.3183159 STANDARD
1 Q0 AP880502-0187 4 1.3060037 STANDARD
1 Q0 AP880921-0032 5 1.2953771 STANDARD
1 Q0 AP880611-0042 6 1.2422574 STANDARD
1 Q0 AP881211-0022 7 1.2422574 STANDARD
1 Q0 AP880226-0124 8 1.1870179 STANDARD
1 Q0 AP880815-0061 9 1.1780995 STANDARD
1 Q0 AP880228-0064 10 1.1605673 STANDARD
1 Q0 AP880320-0059 11 1.143876 STANDARD
```

Lucene has an option with which we can display an explanation of the score calculated.

```
for (ScoreDoc hit : hits.scoreDocs) {
    Document doc = storedFields.document(hit.doc);
    int j=i+1;
    // System.out.println(QNUM + " Q0" + doc.get("docno") + j + " " + hit.score + " STANDARD
    out.write(QNUM + " Q0" + doc.get("docno") + j + " " + hit.score + " STANDARD\n");
    i++;
    Explanation exp = searcher.explain(q, hit.doc);
    System.out.println(exp.toString());
}
```

Let's take for example the top retrieved document **AP881225-044** with a score of **1.7895668**

The score was obtained in the following way:

1.7895668 = sum of:

1.0594481 = weight(text:**overcrowded** in 2310)

[ClassicSimilarity], result of:

1.0594481 = score(freq=3.0), product of:

7.3400717 = idf, computed as $\log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$

from:

140 = **docFreq**, number of documents containing term

79923 = **docCount**, total number of documents with field

1.7320508 = **tf**(freq=3.0), with freq of:

3.0 = freq, occurrences of term within document

0.083333336 = fieldNorm

0.73011863 = weight(text:**prisons** in 2310) [ClassicSimilarity], result of:

0.73011863 = score(freq=2.0), product of:

6.1952615 = idf, computed as $\log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$

from:

442 = docFreq, number of documents containing term

79923 = docCount, total number of documents with field

1.4142135 = **tf**(freq=2.0), with freq of:

2.0 = freq, occurrences of term within document

0.083333336 = fieldNorm

The word "overcrowded" appears in the <TEXT> section of the document 3 times and the word prison appears 2 times.

Lucene calculates the number of documents containing the term and the total number of documents.

We notice that Lucene uses a slightly modified version of the formula for tf and for idf.

For instance, the tf value reported by Lucene is greater than one whereas if you use normalization, the tf value could not be greater than 1.

Also, the idf value is calculated using the following formula:

$\text{Idf} = \log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$

The tf-idf formulas are explained in greater detail in the following documents:

[The Lucene formula: TF & IDF - Ayende @ Rahien](#)

[https://lucene.apache.org/core/8_8_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html#:~:text=tf\(float%20freq\)-,Computes%20a%20score%20factor%20based%20on%20a%20term%20or%20phrase's,initial%20score%20for%20a%20document.](https://lucene.apache.org/core/8_8_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html#:~:text=tf(float%20freq)-,Computes%20a%20score%20factor%20based%20on%20a%20term%20or%20phrase's,initial%20score%20for%20a%20document.)

The Query #25 has the following characteristics:

<top>

<num>25

<title>NRA Prevention of Gun Control Legislation

<desc>

Document will report occasions on which the NRA engaged in actions which prevented the passage of legislation designed to reduce the proliferation of guns.

<narr>

A relevant document will contain specific instances in which gun control legislation was defeated through the efforts of the National Rifle Association. Such instances will include NRA influence to affect a popular vote on proposed legislation as well as the use of gun lobbyists to influence legislators engaged in such voting.

</top>

- ➔ The retrieval was performed using the query title:
NRA Prevention of Gun Control Legislation
- ➔ It produced the following result (top 10)

```
results_tfidf_QTITLE - Notepad
File Edit Format View Help
25 Q0 AP880605-0026 1 3.0951118 STANDARD
25 Q0 AP880606-0019 2 3.047386 STANDARD
25 Q0 AP880427-0150 3 2.9873984 STANDARD
25 Q0 AP880427-0240 4 2.9804544 STANDARD
25 Q0 AP880812-0017 5 2.9740791 STANDARD
25 Q0 AP880811-0163 6 2.9602067 STANDARD
25 Q0 AP880917-0094 7 2.6691823 STANDARD
25 Q0 AP881008-0107 8 2.6447883 STANDARD
25 Q0 AP880426-0221 9 2.6040802 STANDARD
25 Q0 AP880929-0184 10 2.5927305 STANDARD
25 Q0 AP881104-0268 11 2.571032 STANDARD
```

Using the same Lucene specific formulas for the calculating the ranking, the document AP880605-0026 has the highest ranking of 3.0951118.

The search included the following terms: nra (freq=16), gun (freq=11), control (freq=10) and legislation (freq=2).

Lucene calculates the tf (without normalization) and idf using its own adapted formulas.

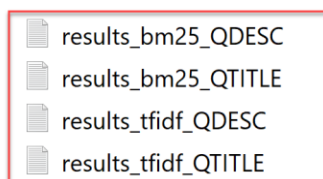
The output displays the top 10 documents ranked in order of relevance for the query 25

3.0951118 = sum of:
1.5827188 = weight(text:nra in 7959) [ClassicSimilarity], result of:
1.5827188 = score(freq=16.0), product of:
8.299848 = idf, computed as $\log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$ from:
53 = docFreq, number of documents containing term
79923 = docCount, total number of documents with field
4.0 = tf(freq=16.0), with freq of:
16.0 = freq, occurrences of term within document
0.04767313 = fieldNorm
0.7491705 = weight(text:gun in 7959) [ClassicSimilarity], result of:
0.7491705 = score(freq=11.0), product of:
4.73817 = idf, computed as $\log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$ from:
1901 = docFreq, number of documents containing term
79923 = docCount, total number of documents with field
3.3166249 = tf(freq=11.0), with freq of:
11.0 = freq, occurrences of term within document
0.04767313 = fieldNorm

0.47853336 = weight(text:control in 7959) [ClassicSimilarity],
result of:
0.47853336 = score(freq=10.0), product of:
3.1742313 = idf, computed as $\log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$
from:
9086 = docFreq, number of documents containing term
79923 = docCount, total number of documents with field
3.1622777 = tf(freq=10.0), with freq of:
10.0 = freq, occurrences of term within document
0.04767313 = fieldNorm
0.28468922 = weight(text:legislation in 7959)
[ClassicSimilarity], result of:
0.28468922 = score(freq=2.0), product of:
4.222624 = idf, computed as $\log((\text{docCount}+1)/(\text{docFreq}+1)) + 1$
from:
3184 = docFreq, number of documents containing term
79923 = docCount, total number of documents with field
1.4142135 = tf(freq=2.0), with freq of:
2.0 = freq, occurrences of term within document
0.04767313 = fieldNorm

Mean Average Precision (MAP)

- Score computed with trec_eval for the results on the 50 test queries.
- Our program was executed for the following combinations of similarity and query title/description. The results are the following:



- We will execute the trec script against each of the above results file
- The trec map results files are located under the results folder.

tf-idf / query title

```
$ ./trec_eval -q -m map C:\\A1_CSI_4107\\queries\\qre150.txt C:\\A1_CSI_4107\\results\\results_tfidf_QTITLE.txt
```

map	1	0.1460
map	10	0.1579

Group 4
11/02/2024

map	11	0.6240
map	12	0.0057
map	13	0.7874
map	14	0.0016
map	15	0.0601
map	16	0.1218
map	17	0.0897
map	18	0.1153
map	19	0.2791
map	2	0.1110
map	20	0.9167
map	21	0.0829
map	22	0.0512
map	23	0.7414
map	24	0.2851
map	25	0.1829
map	26	0.0194
map	27	0.4001
map	28	0.0915
map	29	0.0633
map	3	0.2077
map	30	0.3165
map	31	0.0126
map	32	0.1218
map	33	0.7885
map	34	0.0657
map	35	0.4689
map	36	0.0632
map	37	0.1247
map	38	0.2234
map	39	0.1557
map	4	0.1714
map	40	0.1793
map	41	0.0002
map	42	0.3511
map	43	0.2625
map	44	0.0370
map	45	0.0829
map	46	0.2685
map	47	0.0326
map	48	0.2749
map	49	0.0010
map	5	0.0005
map	50	0.5222
map	6	0.4114
map	7	0.2782
map	8	0.0000
map	9	0.0000
map	all	0.2151

tf-idf / query description

```
$ ./trec_eval -q -m map C:\\A1_CSI_4107\\queries\\qre150.txt C:\\A1_CSI_4107\\results\\results_tfidf_QDESC.txt
```

map	1	0.4323
map	10	0.1186
map	11	0.5651
map	12	0.0088
map	13	0.8715
map	14	0.0356
map	15	0.0456
map	16	0.2090
map	17	0.2082
map	18	0.1368
map	19	0.2561
map	2	0.0185
map	20	0.8056
map	21	0.0299

Group 4
11/02/2024

map	22	0.0403
map	23	0.3822
map	24	0.4665
map	25	0.1315
map	26	0.1981
map	27	0.2635
map	28	0.0619
map	29	0.1702
map	3	0.1498
map	30	0.0731
map	31	0.0469
map	32	0.0447
map	33	0.3743
map	34	0.2032
map	35	0.2218
map	36	0.1100
map	37	0.0340
map	38	0.1049
map	39	0.0208
map	4	0.1445
map	40	0.1341
map	41	0.1236
map	42	0.0976
map	43	0.4463
map	44	0.0324
map	45	0.0887
map	46	0.0906
map	47	0.0138
map	48	0.2089
map	49	0.0090
map	5	0.0071
map	50	0.5453
map	6	0.1061
map	7	0.1151
map	8	0.1000
map	9	0.0000
map	all	0.1821

bm25 / query title (best score)

```
$ ./trec_eval -q -m map C:\\A1_CSI_4107\\queries\\qre150.txt C:\\A1_CSI_4107\\results\\results_bm25_QTITLE.txt
```

map	1	0.1816
map	10	0.1156
map	11	0.7082
map	12	0.0055
map	13	0.9232
map	14	0.0006
map	15	0.2283
map	16	0.3311
map	17	0.0865
map	18	0.1449
map	19	0.2137
map	2	0.1519
map	20	0.9167
map	21	0.1250
map	22	0.0551
map	23	0.7261
map	24	0.4267
map	25	0.1779
map	26	0.0616
map	27	0.5324
map	28	0.1194
map	29	0.1127
map	3	0.3092
map	30	0.2533

Group 4
11/02/2024

map	31	0.0208
map	32	0.3218
map	33	0.8730
map	34	0.0762
map	35	0.4933
map	36	0.1119
map	37	0.1386
map	38	0.2613
map	39	0.1393
map	4	0.1865
map	40	0.1794
map	41	0.0015
map	42	0.3397
map	43	0.3095
map	44	0.0382
map	45	0.1129
map	46	0.2969
map	47	0.0374
map	48	0.2543
map	49	0.0010
map	5	0.0009
map	50	0.6005
map	6	0.4215
map	7	0.4839
map	8	0.0000
map	9	0.0000
map	all	0.2521

bm25 / query description

```
$ ./trec_eval -q -m map C:\\A1_CSI_4107\\queries\\qrel50.txt C:\\A1_CSI_4107\\results\\results_bm25_QDESC.txt
```

map	1	0.5773
map	10	0.1045
map	11	0.6125
map	12	0.0040
map	13	0.9096
map	14	0.0174
map	15	0.0664
map	16	0.5141
map	17	0.2059
map	18	0.1856
map	19	0.2191
map	2	0.0371
map	20	0.8056
map	21	0.0269
map	22	0.0342
map	23	0.3212
map	24	0.5736
map	25	0.1525
map	26	0.2900
map	27	0.1948
map	28	0.0169
map	29	0.2812
map	3	0.1456
map	30	0.0950

Group 4
11/02/2024

map	31	0.0426
map	32	0.0987
map	33	0.3642
map	34	0.4421
map	35	0.2717
map	36	0.1471
map	37	0.0598
map	38	0.1427
map	39	0.0275
map	4	0.1521
map	40	0.0767
map	41	0.1472
map	42	0.0460
map	43	0.4826
map	44	0.0483
map	45	0.0902
map	46	0.0636
map	47	0.0179
map	48	0.3098
map	49	0.0138
map	5	0.0100
map	50	0.5927
map	6	0.1044
map	7	0.2455
map	8	0.2000
map	9	0.0000
map	all	0.2118

Conclusion

Best map score is obtained for the similarity BM25 and the search based on query title.

Task Distribution

Step1. Preprocessing: Hasan Hashim

Step 2. Indexing: Trinity Vermeire

Step 3. Retrieval and Ranking: Maria Stancu

trec_eval script: All