Home Assignment No. 3

Submission date: 07/06/2018

The purpose of this assignment is to gain experience with IR algorithms and learn about evaluation of retrieval results.

This assignment consists of three major parts:

1. Write a program that will run a set of queries on a document collection and produce a ranking of documents.
   This program will have 2 operating modes: basic and improved.
   - The basic mode will retrieve documents using a basic index, query parsing and tf-idf weighting scheme.
   - The improved mode will include any variation you choose that will improve the system's performances on the given benchmark.
2. Perform evaluation on the results of the retrieval process using a given benchmark collection and a query set.
3. Write a report describing your program, the experiments, and your conclusions.
   The report is an important part of your grade. Leave enough time to do a good job on it.

You must submit one .zip, .gz, or .tar file that contains a report and your source code.

Part 1 - the Program

General

Your program should run a single retrieval experiment. A single retrieval experiment is a loop over a set of queries.

For each query:

1. Read one query from the query file.
2. Parse the query.
   You will handle free text queries. Your query parser should handle the following cases:
   1. Discard terms that are stop words – the 20 words that appear most frequently in the collection.
   2. Handle hyphenated terms (e.g., co-linear) and other ambiguous forms of punctuation (use your own judgment about how to handle them).
3. Submit the query: for each query term, fetch the inverted list from the index, if one is available. Note that some query terms may not occur in the index.
4. Sort the matching documents by their tf-idf scores (use a formula at your choice), in descending order, and return the documents with a score above a certain threshold T – those documents are considered relevant. Choose T to be the best you can – try a few values and assess what works best – discuss it in your report.
5. Write the information for retrieved documents to a file, ordered by DocID in ascending order.
6. The described algorithm is the basic algorithm. For the improved algorithm, you should implement a feature that will improve at least one of the measurements in the evaluation. You are free to use any technique that improves retrieval effectiveness.

Source Code

Your software must be written in Java, this is a strict requirement. Although I understand the value of other programming languages, the programming assignments must be done in Java.

You are encouraged to use Lucene, it will make your life easier (but you don't have to).

You should make sure that your source code has reasonable documentation and can be understood by others.

Input

Your software must accept only one parameter which is the name of a parameter file. The parameter file must be located in the current working directory. This parameter file must contain all of the parameters necessary to run your program. Your software must support the following parameters.

- queryFile= the query file full name and path
- docsFile= the documents file full name and path
- outputFile= the file path where your program will write its output
- retrievalAlgorithm= either "basic" or "improved"

An example parameter file is provided (parameters.txt). When testing your software, I will use a parameter file of the same format.

Output

Your software must write the search results to the specified output file in the following format:
QueryID <space delimited list of relevant doc ids>

For example:

```
1 268 288 304 308 323 326 334
2
3 326
```

The QueryID should correspond to the query ID of the query you are evaluating, and the DocID should be the external document ID.

When no documents are retrieved for a query, output a line with the queryID and no doc IDs (like query 2 in the example).

Part 2 - Testing Your Software

Data

Documents (docs.txt)

The corpus is a collection of news articles. All documents are concatenated to one plain-text file in capital letters. Each document begins with the following line:

```
*TEXT <DocID> <Date> PAGE <number>
```

DocID is a unique document number.

For example:

```
*TEXT 018 01/04/63 PAGE 021

RUSSIA WHO'S IN CHARGE HERE ? IT WAS IN 1954 THAT NIKITA

KHRUSHCHEV LAUNCHED HIS GRANDIOSE " VIRGIN LANDS " GAMBLE . PART OF THE

PLAN WAS TO PLOW UP 32 MILLION ACRES OF MARGINAL LAND IN KAZAKHSTAN,

AND SETTLE IT WITH COMMUNIST " PIONEERS, " WHO WERE TO PLANT AND

PRODUCE HUGE QUANTITIES OF DESPERATELY NEEDED GRAIN WITHIN TWO YEARS .
```

There are 423 documents.

Note that there may be redundant newlines, spaces and other characters.

Training queries (queries.txt)

Each query begins with the following lines:

```
*FIND        <QueryID>
```

followed by the query itself, for example:

```
*FIND      1

 KENNEDY ADMINISTRATION PRESSURE ON NGO DINH DIEM TO STOP

SUPPRESSING THE BUDDHISTS .
```

There are 60 queries.
Note that your algorithms will be tested on the given set of testing queries as well as on a different query set.

Relevance judgments (truth.txt)

Specifies which documents are relevant to each query.

The first field is the query number, then a space-separated list of the relevant documents.

Tip

You may find it convenient to test your software on a small index that consists of around 5 very short ("toy") documents. It is very easy to inspect these documents, and to know for certain that your software is doing what you expect. You can prepare such short docs and a short set of queries to test the software first.

The Experiment

You should test your system on the given query set.
For each query you must report Precision, Recall and F score. You should include a table in your report that summarizes your results per query in the “basic” vs. “improved” mode.

Part 3 - The Report

Write a report outlining the decisions you made in implementing the search algorithms, as well as your comments on the relative performance of the techniques you have tried. Provide specific details on the methods and their parameters including formulas and tables. You are allowed a maximum of 4 pages for the report, but shorter reports are fine. Your report should be in Microsoft Word or pdf format. When your uploaded file is unzipped, the report should be in the same directory as your source code.