WRIGHT STATE UNIVERSITY
Department of Computer Science and Engineering

CS7800 Spring 2017 Prasad

Assignment 1 (Due: February 15) (5 pts)

In this project, you will design and implement your own information retrieval system using Lucene APIs. The project has two phases. In Phase I, you will build the index and search application. In Phase II, you will evaluate this search engine along two orthogonal dimensions: (i) quality of results (ranking) and (ii) resource utilization (storage and running times).

The project may be done individually, or in a team of two members. Both the team members are expected to contribute to all aspects of the project: design, implementation, documentation, and testing.

Phase I

Lucene was originally a high performance Java based search engine library. You will use the Lucene API (in Java or Python) to index the documents in Cranfield (and also Medline if you like) to answer some standard queries. Here we describe possible modules you need to complete in Phase I.

Indexer

The indexer will process all the documents in the corpus to create a searchable index. Note that documents in the Cranfield collection are contained in a single file and should be separated into multiple files before indexing. Internally, Lucene stores each document as a set of field-value pairs. Indexer should convert each document file to a Lucene document with fields such as title and abstract. In general, the index can be in-memory or on disk. For this assignment, you are required to write the index to disk which will be utilized by Index Searcher later.

You can try various analyzers to pre-process a document: to tokenize, to remove stop words, and to stem, before indexing. After indexing, you will query the index to retrieve documents based on the query string supplied by a user. Query string has to undergo a similar treatment as document during indexing (that is, remember to use the same analyzers for both indexing a document and query processing).

Index Searcher

Index Searcher takes a user query as input and applies the same set of pre-processing operations applied to the documents during indexing. You can use Query Parser for pre-processing and Index Searcher for finding hits for a particular query. The hits are returned along with the details such as number of hits and details of each hit. Hits should be displayed to the user in a way that allows the user to access the resulting documents. The details of "linking mechanism" and the teaser text to be displayed is left to your creativity.

Enhanced Search Results

- You can selectively boost the importance of fields in a document using Field.setBoost. E.g., you can weigh title more (or less?) depending on your knowledge of the document corpus.
- You can explore other enhancements or experiment with different analyzers, stop word lists, etc.

Milestones for Phase I

- Prepare the corpus according to the input requirements of the Lucene API.
- Implement Indexer to index all the documents in the corpus and store the index on disk (of the local machine).
- Implement Query Parser to process the incoming queries.
- Implement Index Searcher to satisfy the user query and retrieve the relevant documents.
- Enhance the search results and the search engine performance by modifying Lucene API parameters.

Phase I resources

- Lucene API reference: `http://lucene.apache.org/core/`
- Lucene Tutorial: `http://www.lucenetutorial.com/`

Phase I environment setup

After downloading and extracting Lucene API archive, add the following `jar` files to the classpath before proceeding further. If you are using Eclipse, you can copy these `jar` files to a directory within the project. You need to visit project properties and add the copied jar files to the java build path. The location of the relevant `jar` files in the Lucene API download is given below:

- *Core*: PATH_TO_LUCENE/core/lucene-core-6.3.0.jar
- *Standard Analyzer*: PATH_TO_LUCENE/analysis/common/lucene-analyzers-common-6.3.0.jar There are many other analyzers to explore.
- *Query Parser*: PATH_TO_LUCENE/queryparser/lucene-queryparser-6.3.0.jar

Phase II

Compare the quality and the performance of the search engine built in Phase I of this assignment.

Step I

Discuss the evaluation metrics used. Show the effectiveness of your enhancements on search results using the metrics.

Step II

Determine the (i) quality of results (ranking) and (ii) resource utilization (storage and running times) of the search engine you built using the Lucene API.

Benchmarks for Phase II

- *Quality of results*: Compute appropriate metrics for various user queries (at least 5 queries) from the Cranfield collection, and report the evaluation metrics for the test set.
- *Performance/Resource utilization*: (a) Record the time it takes for a user to enter a query, search index, display results, and fetch the relevant document. (b) Note the index size on disk.

Deliverables

You will turn in the following *three* items in the form of a zip-archive: `asg1.zip`.

Lucene experience: Briefly describe your Indexer and Index Searcher built using the Lucene API. Include search enhancements you made along with justification for its effectiveness.

Benchmark output: Present the results of five queries in the benchmark, preferably in the form of a table or any other form suitable for convenient visualization.

Code and accompanying documentation: Include well-documented source code for the entire project. Your documentation must explicitly highlight parts of code for enhancing the search results w.r.t. the baseline.

What to upload?

Upload the archive `asg1.zip` onto `Pilot $>$ Dropbox $>$ Assignment 1` folder.

You are also expected to demo your program to us (the grader and, if necessary, to me) and be prepared to answers questions about its design, implementation, and evaluation. *Plagiarism will be penalized with an award of 0 and further action.*

No deadline extension will be given. You will lose 50% of the points for every day you turn in late.