Web Search Engine
CS172
Part B: Indexing and Searching

Umesh Moghariya, sid: 861083538 Kevin Ng, sid: 860929456

June 7, 2013

1 Overview of the System

The project is an extension of Phase 1. The saved files from the Phase 1 is used as an Input to Phase 2. The saved html pages are indexed. The generated index is then used as an input to Query String. The Query is compared with the Index and documents are retrieved based on the Ranks.

1.1 Architechture

The project uses Apache Lucene to Index and Search the Crawled Web Pages. It then allows to perform queries on this index, returning results ranked by either the relevance to the query or sorted by an arbitrary field as desired by overloading the class.

1.1.1 Indexing

The Indexer creates a special type of Index based on the Crawled Web Pages. This type of index is called an inverted index. An Inverted Index is a word-pages representation of page - words.

The indexing allows various fields to be indexed, tokenized, or just saved depending upon the requirement of the project. In this project we are using:

1. URL - Not tokenised, only Saved.
2. Html Content - Saved, Tokenized, Indexed.
3. Document Name - Not Tokenized, Only Saved.

1.1.2 Searching

Searching takes an index as an input. It involves creating a Query via a QueryParser and hands this Query to an IndexSearcher, which returns a list of Hits. Hits are based on:

1. Words matching in Documents.
2. Words repeating in Documents.
3. Frequency of Words through out the Document.

1.1.3 Scoring

The project uses the default scoring model of Lucene which is a TfIdf scoring model. Scores are influenced depending upon:

1. tf
2. idf
3. coord

4. lengthNorm
5. queryNorm
6. boost (index)
7. boost (query)

In this project we have not used Boost, so equal weights are applied to all the terms. The Scoring function ranks the retrieved Documents for a given Query. And the results are displayed in Decreasing order of the ranks.

1.1.4 Documents, Index

In Lucene, a Document represents search and index. Many documents can be indexed. Indexing adds Documents to an IndexWriter, and searching involves retrieving Documents from an index via an IndexSearcher.

2 Limitations

1. Query parsing for and - removes the character but does not inseart space.
2. The Search and Index functions were together and hence each query would index again.
3. For Snippet generation, tokens are not highlighted.

The first two points have been corrected and implemented.

3 Deployment Directions

Attached as a Readme file.

4 Screen Shots

Check Folder named as ScreenShots.

5 Features

1. 8000 saved pages.
2. Duplicate Pages Detection.
3. Snippet generation.
4. Advanced Search for using Lucene WildCards.

6 Collaboration Details

1. Umesh :
   (a) Logic and Program Flow.
   (b) Datastructure Design.
   (c) Code for Indexer and Search.
   (d) JSP servlet implemetation.
   (e) Snippet Generation.
   (f) Report Writing.
2. Kevin

(a) Logic and Program Flow.
(b) Datastructure Design.
(c) Check for duplicate content.
(d) Parsing Search Queries.
(e) Testing.