Polytechnic Institute of NYU

Computer Science and Engineering

CS 6913, Spring 2013

Jiankai Dang, Poly #:

Zhuoran Yu, Poly #: 0489525

Assignment #3 Web Search Engine Report

1. **How to Implement a Search Engine: Crawl Web Pages**

*Building Web Crawler is assignment #1. It is somehow relative to this assignment. So in Part 1, we will briefly talk about crawler, and then talk about how it connects to this assignment.*

* 1. **Crawler**

Crawler, also named spider, is a program to collect data on the web. It tries to download all files it could research on the web site and store.

* 1. **Data**

Crawler downloads pages. Save a number of pages in one file. Compress these files and store them. The data given to us for assignment 3 is not the data set we crawl in Assignment #1. Due to this reason there are some features we want to do but cannot in this assignment.

* 1. **More about Data: Duplicate Contents**

In assignment #3, we add a feature that program check duplicate contents in result set. If the load is huge, it is not an efficient way to do duplicate detecting in query processing part. Instead we should detect duplicate contents when store files. If so, the duplicate detecting in query processing part would be more efficient.

* 1. **More about Data: Update**

In practice, a search engine should not be “static”. The crawler is always downloading web pages, adding them into storage. At the same time, we should always be updating our index and show latest result to customers. In this assignment, we could not do this since the data set is static.

1. **How to Implement a Search Engine: Build Index**

An inverted index, also named posting files or inverted files, is an index data structure storing mapping from each terms to content include it. For example, words (or word id) to its location in a set of documents or in a database files. The purpose of inverted index is to fast full text searches. [1]

For example, Document 1 is [“Inverted Index”], Document 2 is [“Inverted Index is a kind of Index”]. The inverted index of word “Index” is [[1, 1], [2, 2]], meaning that this word is in document 1 for 1 time and in document 2 in 2 times.

* **Build Index**

This part is within assignment 2, we would introduce here very briefly. Suppose the memory is large enough, we could scan all files and add document id into inverted index of terms within the document. Unluckily, in practice memory is probability not enough. Our solution is to divide files into several parts. We build up inverted index separately for each part. Finally, our program uses an I/O efficient algorithm to merge these inverted indexes (in most of time, the algorithm is merge sort). We will introduce some useful or highlights details of our project

* Parsing the Collection

When building index, we should make a decision whether build index for non-natural word, such as “0fxxx”. If not, we could use Natural Language Toolkit [2] to filter. This would make inverted index in a small size. The speed would be highly increased. If so, our user could get result of special search queries. For example, someone want to search a special line of code “string.rfind(x)”. However, the inverted index would be much larger. In this project, we choose a third way. We do build inverted index for every term. For efficient, we build up a cache for common words in natural language. This would be told later in details.

* Efficient Building
* Index Compression

1. **How to Implement a Search Engine: Query Processing**
2. **About Programs: High-level Structure**
3. **About Programs: Modules**
4. **About Programs: Files in details**
5. **Search In Action: How to Run it**
6. **Search In Action: Experiment**
7. **Search In Action: Limitation:**

**Reference:**

[1] <http://en.wikipedia.org/wiki/interted_index>