Web Information Retrieval (67782)
Ex2: Index Construction

1 Exercise Description

In this exercise, your goal is to improve the time needed to construct your index, and to allow it to scale up to very large datasets. Thus, you will be submitting the following class, along with classes used to read from your index (e.g., ReadIndex from Exercise 1).

```
package webdata;

public class IndexWriter {

    /**
     * Given product review data, creates an on disk index
     * inputFile is the path to the file containing the review data
     * dir is the directory in which all index files will be created
     *     if the directory does not exist, it should be created
     */
    public void write(String inputFile, String dir) {}

   /**
     * Delete all index files by removing the given directory
     */
    public void removeIndex(String dir) {}
}
```

You should no longer assume that the raw data can fit at once in main memory. Instead, you should use algorithms discussed in class (or develop your own) to allow index creation despite the large size of the input. You will probably use the index structure that you developed in Exercise 1, but you can change this structure, if you so desire.

Note also that all operations of IndexReader should remain reasonably efficient, regardless of the size of the dataset that you have indexed. If you properly designed the IndexReader in ex1, then this will naturally occur. Otherwise, you will need to make changes to IndexReader.

Your program will have **1GB of heap space** when running, i.e., we will run your program with the flags `-Xmx1G -Xms1G`. You should also use these flags while developing and testing your solution.

2 Experimental Data

Experimental analysis of your system should be carried out. Experiments should be run with the Movie and/or Book review data from the Stanford Large Network Dataset (snap). A link to these datasets will be provided.

Your experiments should be run by starting to index 1,000 reviews, and then with a number of reviews that increases by multiples of 10, until indexing the entire dataset(s), or reaching a dataset size for which your program can no longer complete index construction, within reasonable time, i.e., approximately 2 hours. If you can successfully index the entire Movie Review dataset, you should continue trying to index with increasing review sizes by appending (parts of) the Book Review dataset. So, for example, you might provide experimentation with datasets containing 1,000 reviews, 10,000 reviews, 100,000 reviews, 1,000,000 reviews, and the entire Movie Review dataset (and then even larger data sets if possible).

2.1 Analysis Requirements

In addition to handing in the code, you should hand in an experimental analysis of the

- runtime for index creation
- the size on disk that the index requires and
- the time it takes to run 100 random requests for getReviewsWithToken, and the time it takes to run 100 random requests for get TokenFrequency.

This analysis should consist of graphs showing runtime / index size, as a function of the number of reviews in the index.

The runtimes of your experimentation will, obviously, be dependent on the system used for testing. Therefore, you should state, in your written analysis, the type of computer used for testing. You can use the computers available at huji, or any other computer available to you. If you use one of the huji computers, it is sufficient to state the computer name. For other computers, state in your report important system parameters: operating system type, CPU speed, size of internal memory.

2.2 Exercise Submission

The code for your exercise should be submitted in a jar format via the course website. Only one submission should be made for each pair! Please make sure that the jar file uploaded is in the correct format, and contains all relevant files, including: all source files, the file analysis.pdf containing your experimental analysis, a README (including any compiling or running issues, and the logins and teudat zehut numbers of the students submitting the project). You jar file should include your implementation of IndexReader.

Your exercise should run correctly both on Linux and on Windows systems.

3 Additional Remarks

Although we do not state precise scalability requirements in this document, your system will be compared to that of other students in the course. The grading of this exercise will take into

consideration scalability of the system (in terms of time to index, index size, and access time via the methods of IndexReader) and points will be given based on the relative scalability of the system.

The programming itself, as well as the experimental analysis can be time-consuming. I highly recommend that you get started early.

We are aware that due to memory constraints in huji computers, it may be difficult to work with very large datasets. Please use the "tmp" directory, as needed, for running your program. Keep in mind that there is no backup for this directory. If necessary, email the course staff, and we will check into increasing your storage space for the duration of the course.

3.1 Frequently Asked Questions

How do I know the block size on my computer? How can I determine how many blocks can be simultaneously read into main memory? You cannot easily do this. However, it is also not very important to precisely determine these values. As needed, choose default values for these that make sense (stored in constants), and then either make them larger (if your program is too slow), or smaller (if you crash on an out of memory exception).

How many intermediate files can I create? There is no precise limit on the number of intermediate files. But, you should not be creating a huge number of files, e.g., a file per posting list. Such an implementation will be much too slow, as it will require your operating system to work quite hard.

What do I do if I find something strange in the input, e.g., the helpfulness ratio is greater than one? That is the unfortunate reality in real data. You can choose any reasonable way to recover from this, without skipping over any of the reviews. For example, if your helpfulness ratio is 8/1, you can store a score of 1/8.