Assignment 1: Text Statistics

Subject: Advanced Java

Semester: Fall 2019

Student: Niko Tili

Instructor: Elton Ballhysa

Contents

Introduction

The program is used for computing text statistics. It processes some text files by calculating and displaying following statistics in runtime: number of processed files, number of processing files, five most frequent letters, five most frequent letter pairs, five most frequent words, unigram entropy, bigram entropy, total words and standard deviation of words. The whole processing is done concurrently.

Solution

The user should specify the folder path (if no path is specified the program exits with an error message). The path is passed to ***Controller***. ***Controller*** (extends ***java.lang.Thread***) is a final class, from which is created only one instance. This thread priority is set to ***Thread.MAX_PRIORITY***. It deals with:

- Starting the reading procedure
- Loading stop words
- Calculating the proper number of threads for the reading procedure
- Printing elapsed time (every 500 ms)
- Printing the number of processed files so far (every 500 ms)
- Printing the number of still-processing files (every 500 ms)
- Printing most frequent ***Sequences*** and their frequency (every 500 ms, final result)
- Printing unigram and bigram entropy (every 500 ms, final result)
- Printing the total execution time (final result)
- Printing the total processed files (final result)
- Printing the total word count (final result)
- Printing the standard deviation of words (final result)
- Terminating the program when finished

If the path is not valid (does not exist or is not a folder) the program exits. After verifying the path is valid, a Stream containing paths of all .txt files on that folder is created and for each path a single ***FileConsumer instance***, consumes each path as following:

- A thread from a fixed thread pool is started and executes ***loadFile(Path path)***
- In the ***loadFile(Path path)*** method, the text file is split into words
- Special characters are removed from each word
- Words are mapped to ***Word*** (extends ***Sequence***)
- For each word unigrams and bigrams are extracted and saved
- If word is not a stop word it is saved too
- Number of words for that file is saved
- Number of processed files is updated

The size of ***THREAD_POOL*** is set by ***getProperNumberOfThreads() Controller*** method. All extracted data, are saved in a single ***SharedRepository instance***, which contains:

- ***unigramMap*** (contains each unigram and its frequency)
- ***bigramMap*** (contains each bigram and its frequency)
- ***wordMap*** (contains each word and its frequency)
- ***fileWordCountMap*** (contains each filename and the specific number of words on that file)

The ***SharedRepository class*** also contains:

- ***computeAndGetStandardDeviationOnWordCount()*** method (computes and returns the standard deviation of word counts among text files)
- ***computeAndGetUnigramEntropy()*** method (computes and returns unigram entropy)
- ***computeAndGetBigramEntropy()*** method (computes and returns unigram entropy)
- synchronized ***putIn*** methods (used to put words, unigrams, bigrams to their corresponding map)
- ***getWithLimit*** methods (used to get most frequent words, unigrams, bigrams and their corresponding frequencies from their maps)
- ***getCurrentTotalWordCount()*** method (used to get the current total word count)

Classes ***Word***, ***Unigram***, ***Bigram*** (inherit from ***Sequence***) represent words, unigrams and bigrams. ***Sequence*** class is an abstract class with an abstract method ***validate(String value) throws ClassCastException.*** It throws an ***Exception*** if the ***Sequence*** is not valid. It was helpful in debugging and producing correct results. Its functionality is controlled by ***SEQUENCE_VALIDATION_ENABLED*** (***false*** by default) found in ***Controller*** class. ***Word*** class contains ***getUnigramStream()*** and ***getBigramStream()*** methods, which extract unigrams and bigrams from a particular ***Word***. ***MapEntryComparator*** (implements ***Comparator<Map.Entry<? extends Sequence, AtomicLong>>***) is used to sort ***Sequence Maps*** by the frequency in ***getSequencesFromMapWithLimit*** method.

Assumptions

- The folder cannot contain more than 1000 .txt files
- JDK 1.8 or later.

How to execute

java al.unyt.edu.advjava.fall2019.assign01.TextStatistics <path> to start the program. The user should specify the folder path as the first argument. If zero arguments are provided the program exits. If more than one argument is provided, only the first one is considered.

Notes

About ***Controller.getProperNumberOfThreads()*** method. It returns the number of processors available to the JVM minus two because one thread is the ***main-thread*** of the application and one is the ***Controller.getInstance()***, still I am not 100% sure about this approach. I think that, the number of threads created in this application should not be the same for different computer systems.