Haluk Dogan 66063720
ELEC865 — Spring 2014 **Homework 2** Due: February 14, 2014

Problem 1

For this assignment, we found two text datasets which contains only English alphabet and punctuation marks. In order to be ensure that they have the same type of characters, we convert them to lowercase. The two files contain 3982 and 3819 characters respectively. We tested the Java library we found by first compress one of the file, and then decompress it. We verified the compression and decompression by using Unix command *diff*. The *diff* command gave zero difference by comparing original file and the one after we obtain decompression. We also use Unix command *wc* to count characters in the file. In the first two types of analysis we use *Huffman coding*, and in the last one we use *adaptive Huffman coding*.

Analysis

- **Analysis I:** In the first type of analysis, we compress the two files with their own code tree. Compression ratio for the test files are [3982 : 2388] (1.67) and [3819 : 2310] (1.65) respectively. We obtain the original files exactly the same after the decompression of compressed files.
- **Analysis II:** In the second type of analysis, we compress the files with the code tree of other file. Compression ratio for the test files are [3982 : 2390] (1.67) and [3819 : 2314] (1.65) respectively. However, after the decompression, we couldn't obtain the original files. We use the *Levenshtein Distance* to evaluate how close the original file and the one after decompression. The *Levenshtein Distances* between original files and the one after the decompression are 327 and 350 respectively.
- **Analysis III:** In the third type of analysis, we compress the files and their compression rates are [3982 : 2346] (1.7) and [3819 : 2271] (1.68). We verified that we obtain the original files exactly the same after decompression of compressed files.

Discussion

In Huffman coding, we use a fixed coding table which is calculated by frequency distribution of the characters to be coded. In adaptive Huffman coding, coding table is generated individually from the character probabilities for the data to be coded. Although adaptive Huffman coding results in higher compression ratios, all the data to be compressed must be available before compression starts because character frequencies should be determined before we can determine the codes. In Huffman coding, if we use different coding tree other than the proper one, we get a different coding.

Appendix

Java Library: `https://github.com/nayuki/Huffman-Coding`
My Files: `https://github.com/haluk/data-compression`