Data Analysis, Homework 3

Elizabeth Gould

June 1, 2025

1 Compressor Statistics

1.1 Issues

I believe the cycles constant at the end are not recorded properly. Everything else appears to be acceptable.

1.2 Texts

I have 4 texts with which I am working with, after which I need to try something more difficult.

1. With This Ring Season 1: Text has been cleaned. Size is 4115 kB on disk.
2. With This Ring Season 2: Text is uncleaned. Size is 5632 kB on disk.
3. With This Ring Season 2: Text still uncleaned, but one section has been removed. Size is 5632 kB on disk.
4. With This Ring Season 1 + 2: This is just text 1 and text 3 combined into one file. Size is 9746 kB on disk.

1.3 Static Machines

Compressor as of right now only works on the first text. It should be easily alterable to work on the other texts, but I would need to clean the second text, and there is an issue with time requirements.

- k = 3
- window size = 40 bits
- Decapitalization time: 1.5 hours
- Decapitalization size: 24 kB
- find count time: 1.5 hours, of which 20 min actually unnecessary

- find count size: 89 kB on disk
- encryption time: 3 hours, but less for an alternative counter
- encryption size: 1012 kB
- full encryption size: 1170 kB?
- full decryption time: 70 min almost all of which are decryption

The long time here compared to the short time for the dynamic code indicates an inefficiency in the code. I have very little desire to find it, but the only difference is how the counts are stored, so it seems a recursive storage is best. I believe I can search the code in a single pass for better efficiency for decapitalization and find counts.

The old method for storing the encoder array took 20 minutes to encode. I think searching the long arrays is inefficient. If I need to rerun this, I will try to clear the inefficiencies.

2 Comparison to Static Machines

For text 2, N = 64 too small. I found a chunk of text strangely written (ultra rare). I think I can do something to deal with this, but I am not sure on my energy level right now. I deleted the problematic part of the text and tried again. However, I think it is possible to come up with an alternate workaround (or maybe one of the alternatives works).

For text 4, as I suspected, the decoder reaches a problem at the first non-ascii character.

- k = 3
- window size = 64 bits
- encoder time, text1: 3 min
- decoder time, text1: 4.5 min
- size text1: 1181 kB
- encoder time, text3: 5+ min
- decoder time, text3: 6-10 min
- size text3: 1575 kB

3 Update Exclusions

So far, doesn't seem faster, but might save some on size. It bypasses the problem with decoding ultra-rare cases, but for text2, I encountered the end of file issue. Manually telling it to decode the last four characters returns the original text, so it is just stopping at the wrong place. Text1 has an end of file issue of 1 character.

The decoder is still failing at the first non-ascii character for text 4.

- k = 3
- window size = 64 bits
- encoder time, text1: 3 min
- decoder time, text1: 5 min
- size text1: 1165
- encoder time, text2: 4.5 min
- decoder time, text2: 8 min
- size text2: 1557 kB
- encoder time, text3: 4.5 min
- decoder time, text3: 7.5 min
- size text3: 1556 kB

4 Choice of First Model

5 Code

5.1 File Format

5.2 Encoder

5.3 Decoder