Data Analysis, Homework 3

Elizabeth Gould

June 17, 2025

1 Compressor Statistics

1.1 Intro

- Escape probability given by C.
- In an old version of the code, if there is an extremely rare character or context, the character will not be correctly read, after which the decoder will soon encounter an error. I have fixed this problem, but many of the statistics use the old decoder.
- The number of cycles where the last bit doesn't change at the end was not read properly. I believe this has been fixed, but it doesn't effect statistics in any case.
- There is a new wrapper code. They will all give results with two more bytes (k and N) than the statistics for which I have here. I just hadn't thought about sending these two numbers.
  - encode.run_encoder(import_file, export_file, k = None, N = 64)
    * import file and export file are file names
    * k = None will try to calculate k based on the most likely character and context, otherwise k gives the desired context length
    * N is the length of the window, and must be a whole number of bytes
    * update_exclusion = False, implements update exclusion variant when True
    * first_model = False, implements choice of the first model when True
  - encode.run_decoder(import_file, export_file)

1.2 Texts

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.
5. With This Ring Season 1: Decapitalized version of the text. It is actually not saved to disk, as it is easy to produce in Python.
6. With This Ring Season 1: Reversed version of the text. It is actually not saved to disk, as it is easy to produce in Python.
7. With This Ring Season 1: Decapitalized and reverse version of the text. It is actually not saved to disk, as it is easy to produce in Python.

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 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.

1.4 Dynamic Machines without Masking

For text 2, N = 64 too small. I found a chunk of text strangely written (ultra rare). I deleted the problematic part of the text and tried again.

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

1.5 Dynamic Machines with Masking

In this section, I added masking of characters which we already ruled out when we choose to encode the new character. So far, it 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. I stopped looking for this afterwords. The data is in Table 1. Table 4 updates the timing.

The decoder is still failing at the first non-ascii character for text 4. This may require the second fix which I noted in the issues section. For all except k = 3, text2 also fails. I have updated the decoder, so the statistics for text2 use a different decoder than the others, which works for text2.

k = 6 appears to be the best variant from the table, but k = 5 is not much worse and faster. This contrasts with the k = 4 estimate for the recommended length of k.

1.6 Update Exclusions

This is for only updating the accessed data points. Results appear slightly better that before, but not by much. The data for this is in Table 5. This table has runs with a newer version of the code than Table 4, and therefore the timings are not completely consistent. The byte size is also not completely consistent, as they have 2 more bytes at the beginning due to sending information of model parameters.

Table 1: Encoding and decoding with masking time and space requirements based on text and maximum length of context. Order is – number of bits, encoding time, decoding time. Note that text 2 uses an updated decoder, so the times are not based on the same metric. Also all the timings are unstable.

	Text 1	**Text 2**	**Text 3**	**Reverse**	**Decap**	**Decap Rev**
find_k k	k=4	k=4	k=4	k=8	k=4	k=8
k=2	1476416 3 m 23.8s 5m 23.1s	1990550 4m 32.5s 10m 4.9s	1990215 4m 57.6s 9m 7.4s	1476623 2m 52.5s 5m 11.9s	1470545 3m 14 s 4m 37.9s	1470470 3m 32.9s 7m 28.3s
k=3	1192730 3 m 9.2s 4m 50.1s	1593626 4m 13.5s 9m 29.2s	1593289 5m 29.2s 7m 14.1s	1193200 3m 6.0 s 5m 3.6 s	1190498 2m 48.7s 4m 28.7s	1190568 4m 12.8s 6m 27.3s
k=4	1058909 3m 36.6s 5m 29.3s	1399465 6m 1.2s 10m 54.4s	1399128 6m 6.4s 9m 46.1s	1059650 3m 55.0s 6m 6.1s	1057195 3m 25.5s 5m 4.1s	1057632 4m 54s 7m 20.7s
k=5	1017021 4 m 22.6s 6m 2.8s	1335544 5m 46.9s 8m 56.5s	1335207 7m 32.5s 10m 49.4s	1018142 5m 1.3s 7m 20.3s	1012264 3 m 51.6s 5m 17.3s	1013437 6m 21.1s 6m 47.7s
k=6	1012453 6 m 10.6s 6m 34.5s	1325093 7m 10.2s 10m 30.9s	1324756 8m 27.2s 10m 11.2s	1013572 6m 11.2s 8m 32.9s	1004875 5 m 24.5s 6m 43.4s	1007044 5m 38.3s 7m 1.8s
k=7	1023743 6m59.5s 8m 8.7s	1338156 9m 91.s 11m 52.8s	1337819 9m 49.2s 12m 15.4s	1024888 7m 38.6s 9m 12.8s	1014294 6m 11.5s 7m 43.6s	1018111 6m 37.7s 8m 2.0s
k=8	1040281 8m 14.0s 10m 29.9s	1359464 11m 59.5s 14m 30.2s	1359126 12m 55.5s 14m 58.7s	1041179 8.5m error	1029459 7m 47.5s 8m 50.4s	1034963 7m 48.6s 8m 36.4s

Table 2: Statistics for the 100MB Wikipedia benchmark.

Variant	k	Size (%)	Encode Time	Decode Time
Standard	5	24.33%	41 m 14 s	66 m 40 s
Standard	6	23.48%	50 m 36 s	73 m 57 s
Update Exclusion	5	24.10%	27 m 20 s	52 m 34 s
Update Exclusion	6	23.26%	31 m 38 s	55 m 17 s
First Model	6	25.64%	169 m 25 s	186 m 45 s
First Model	7	24.92%	187 m 00 s	200 m 57 s
First Model	8	24.45%	208 m 02 s	217 m 07 s
Both	6	23.69%	142 m 37 s	156 m 57 s
Both	7	23.33%	150 m 14 s	162 m 01 s

1.7 Choice of First Model

I have found no variant of interpretation of this idea helpful. It always increases the time requirements (and ensures that encoding time is similar to decoding time, although both are greater than the old decoding time). Statistics for the best interpretations are in Table 5.

Variants:

- Choosing model based on the full statistics. The preferred k is definitely not optimal.
- Choose character with greatest probability (at the root when there is no update exclusion) and find the greatest probability for it at every machine length. Results for text 2 for k = 6 are 2851636 (if not in set, p=0), 2815064 (if not in set p=0 except for the smallest such case, for which p=p_new), 2367140 (if not in set, p = p_new). All are significantly greater than the optimal 1324595 for this text and k.
- Calculate for every step, for every potential machine length, which one has the greatest probability for the most likely character of that machine length. The best performing variant is with update exclusion, which is slightly worse than with or without update exclusion without the first model choice. Without update exclusion, updating the full tree every time, is worse, but not by much. k prefers to be larger in this case than for other cases. All other variants are noticeably worse preforming.

2 Wikipedia Benchmark

Statistics for 100MB are in Table 2, and statistics for 1 GB are in Table 3.

Table 3: Statistics for the 1G Wikipedia benchmark.

Variant	k	Size (%)	Encode Time	Decode Time
Standard	5	21.81%	385 m 37 s	616 m 14 s
Standard	6	20.30%	522 m 42 s	693 m 55 s
Update Exclusion	6			
First Model	8			
Both	6			

Table 4: Encoding an decoding time based on text and maximum length of context on a faster computer for comparison with Wikipedia benchmark times. Order is – encrypted size in bytes, encoding time, decoding time. Text 4 has an estimated k of 4.

k	**Text 1**	**Text 2**	**Text 3**	**Text 4**	**Rev**	**Decap**	**Dec+Rev**
2	1476416 55.2s 2m 14.8s	1990550 1m 18.3s 3m 9.4s	1990215 1m 16.9s 3m 8.5s	3487140 2m 19.4s 5m 43.9s	1476623 53.5s 2m 5.9s	1470545 54.1s 2m 10.2s	1470470 55.9s 2m 10.3s
3	1192730 1m 0.9s 2m 0.2s	1593626 1m 24.4s 2m 44.1s	1593289 1m 24.6s 2m 45.7s	2789743 2m 33.3s 5m 6.9s	1193200 58s 1m 54.2s	1190498 58.6s 1m 55.2s	1190568 1m 0.2s 1m 58s
4	1058909 1m 12.5s 2m 3.6s	1399465 1m 39.3s 2m 49.1s	1399128 1m 40.5s 2m 49.2s	2438273 4m 34.9s 7m 54.8s	1059650 1m 8.8s 1m 57.4s	1057195 1m 8.5s 1m 56.8s	1057632 1m 11.2s 2m 3.1s
5	1017021 1m 28.4s 2m 15.4s	1335544 1m 59.1s 3m 3.9s	1335207 1m 59.3s 3m 2.6s	2311383 5m 28.8s 6m 11.8s	1018142 1m 22.9s 2m 9.0s	1012264 1m 22.9s 2m 7.4s	1013437 1m 25.8s 2m 13s
6	1012453 1m 49.5s 2m 32.4s	1325093 2m 27.5s 3m 25.8s	1324756 2m 24.7s 3m 25.1s	2280430 4m 22.2s 6m 11.9s	1013572 1m 42.9s 2m 25.9s	1004875 1m 42.6s 2m 26.8s	1007044 1m 47.4s 2m 32s
7	1023743 2m 13.2s 2m 55.8s	1338156 3m 0.1s 3m 55.4s	1337819 2m 59.8s 3m 57.2s	2294026 5m 20.1s 6m 57s	1024888 2m 8.6s 2m 48.3s	1014294 2m 7.9s 2m 47.4s	1018111 2m 14s 2m 54.6s
8	1040281 2m 44.4s 3m 22.7s	1359464 3m 38.2s 4m 32s	1359126 3m 37.3s 4m 30.3s	2326546 6m 24.2s 8m 2.2s	1041179 2m 38.1s 3m 14.7s	1029459 2m 36.4s 3m 13.8s	1034963 2m 45.1s 3m 21.1s

Table 5: Encryption size, and encoding and decoding time based on text and maximum length of context for update exclusion and first model variants. Order is – encrypted size in bytes, encoding time, decoding time. k from 5 to 7 for update exclusion, 6 to 9 for first model, and 5 to 7 when both variants are used together. UE = update exclusion, FM = choice of the first model, B = both

Type	**Text 1**	**Text 2**	**Text 3**	**Text 4**	**Rev**	**Decap**	**Dec+Rev**
UE5	1014029 1m 08s 2m 02s	1330437 1m 27s 2m 37s	1330146 1m 31s 2m 44s	2302544 2m 32s 4m 37s	1013960 1m 04s 1m 54s	1009975 1m 04s 1m 56s	1010582 1m 02s 1m 53s
UE6	1013181 1m 23s 2m 15s	1324595 1m 46s 2m 52s	1324304 1m 51s 3m 00s	2279118 3m 01s 5m 00s	1013191 1m 18s 2m 08s	1006058 1m 19s 2m 10s	1007908 1m 16s 2m 05s
UE7	1029213 1m 45s 2m 36s	1343801 2m 11s 3m 16s	1343511 2m 18s 3m 25s	2303153 3m 43s 5m 38s	1029317 1m 40s 2m 26s	1020037 1m 40s 2m 28s	1023760 1m 37s 2m 24s
FM6	1055325 7m 03s 9m 10s	1388465 12m 28s 13m 38s	1388150 7m 47s 8m 43s	2419931 13m 07s 15m 05s	1058296 5m 05s 5m 33s	1048529 4m 41s 5m 15s	1055152 4m 32s 5m 09s
FM7	1041703 9m 36s 10m 13s	1368991 14m 15s 15m 08s	1368667 8m 59s 9m 38s	2376351 15m 04s 16m 41s	1043577 5m 49s 6m 13s	1034037 5m 31s 5m 55s	1038521 5m 21s 5m 47s
FM8	1039151 11m 11s 11m 22s	1363879 16m 22s 16m 48s	1363552 10m 20s 10m 42s	2357563 17m 24s 18m 26s	1040216 6m 44s 6m 57s	1030570 6m 28s 6m 38s	1034643 6m 18s 6m 30s
FM9	1043447 13m 02s 12m 39s	1367958 18m 46s 18m 27s	1367627 11m 49s 11m 48s	2357561 19m 52s 20m 20s	1044944 7m 48s 7m 42s	1033898 7m 31s 7m 24s	1039542 7m 18s 7m 16s
B5	1025369 3m 55s 4m 32s	1347516 5m 43s 6m 36s	1347221 5m 47s 6m 41s	2336603 9m 40s 11m 34s	1025601 3m 37s 4m 21s	1022496 3m 28s 4m 05s	1021005 3m 18s 3m 57s
B6	1017280 4m 32s 5m 03s	1331768 6m 37s 7m 23s	1331499 6m 39s 7m 24s	2295518 11m 04s 12m 53s	1019168 4m 17s 4m 50s	1011298 4m 03s 4m 34s	1012638 3m 53s 4m 26s
B7	1026642 5m 17s 5m 39s	1338470 7m 38s 8m 12s	1338182 7m 36s 8m 09s	2303394 12m 59s 14m 25s	1026813 5m 00s 5m 24s	1018636 4m 46s 5m 08s	1020386 4m 34s 4m 59s