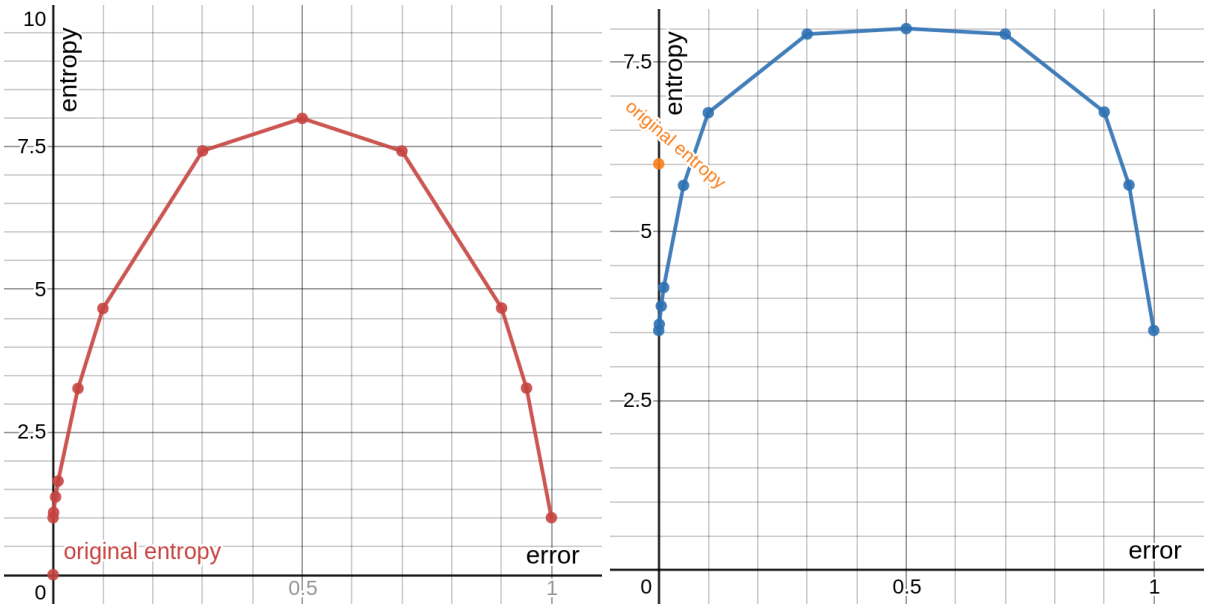
Hersh Rudrawal
hrudrawa@ucsc.edu
05/9/2021

CSE13s Spring 2021
Assignment 5 Writeup

The following are the graphs of entropy produced by the files aaa.txt and random.txt.

Aaa.txt entropy/ random.txt entropy




The entropy produced by the files is marked original entropy. The rest of the graph shows the entropy of the files when encoded in Hamming(8,4). The x axis is the %(1=100%) of error introduced in the encoded file.

Unsurprisingly the first file had low entropy since it only consisted of the letter ‘a’ repeated. And when encoded that entropy increased but stayed relative low. This is because when encoding we are taking a byte, splitting it into 2 halves and encoding each half. As a result, when we encode the letter ‘a’- we end up with 2 different characters. Since the encoded file consists of only 2 different characters, the resulting entropy is 1. As we introduced errors into the encoded data, the entropy began to rise, surpassing 7.5. But once we started introducing errors every other byte(0.5), the entropy started to decline back to the original value. When every bit is an error(at 1), which means every bit has been flipped , the file will have the same entropy as it started. 100% error is similar to having no errors, but the message will obviously not be the same. So when we pass the 50% error mark, the number error bits start to make up a majority of the file, and the entropy starts to drop.

The second graph shows the entropy of random.txt. This file starts off with a lot of entropy but when encoded that entropy drops. This might be because when we encode the data, we split the bytes into 2, and format the nibble into bytes in a specific way, where certain

bit positions are reserved for data, and others for error handling. Since this structure is consistent for all the bits it may result in lower entropy. When we added errors to the encoded data, the behavior was similar to that of aaa.txt. One thing I did realise was that the entropy did not reach 8, not sure why.