Information Theory Assignment

s1140740

1 Source Coding

1.1 Character Statistics

The entropy $H(X_n)$ is 4.168.

1.2 Bigram Statistics

a. The joint entropy $H(X_n, X_{n+1})$ is 7.572.
b. Because they are not i.i.d.
c. The conditional entropy $H(X_{n+1} \mid X_n) = H(X_n, X_{n+1}) - H(X_n)$ is 3.404.

1.3 Compression with known distributions

We know that the message length is always within two bits of the Shannon information content [^1] and thus the maximum number of bits is computed as:

$$\left\lceil h(x) \right\rceil + 2 = \left\lceil -\sum_{n=1}^{N} \log_2 P(x_n) \right\rceil + 2$$

where x_n is the n-th character in the file and N is the length of the file. Probabilities from question 1 are used.

The maximum number of bits an arithmetic coder would use to represent `thesis.txt` assuming this model is therefore 1,433,836.

The maximum number of bits an arithmetic coder would use to represent `thesis.txt` assuming the more sophisticated model is 1,171,194. The length is computed as:

$$\left\lceil -\log_2 P(x_1) - \sum_{n=1}^{N} \log_2 \frac{P(x_{n+1}, x_n)}{P(x_n)} \right\rceil + 2 = \left\lceil -\log_2 P(x_1) - \sum_{n=1}^{N} \log_2 P(x_{n+1} | x_n) \right\rceil + 2$$

The joint probabilities from question 2 are combined together with probabilities from question 1 to compute the conditional probabilities which are used in chain rule.

1.4 Compression with limited precision header

Assuming we transmit the powers of 2 required to get back the original probabilities, we require 8 bits per character. Since our alphabet is of size 27 (`'a'` - `'z'` and `' '`) we require $27 * 8 = 216$ bits for the header if we send the powers ordered alphabetically with space at the end.

Furthermore, the probabilities are re-normalised before use (by both sender and receiver) to add up to 1 and then applied in the same fashion as in the previous question to compute the maximum number of bits required. Thus we need 1,435,081 bits for data and 1,435,297 bits altogether with this scheme.

TODO!

1.5 Compression with adaptation

Using the Laplace prediction rule for the i.i.d. model the maximum number of bits required to encode `thesis.txt` is 1,434,027.

The rule is computed for every character of the file and these probabilities are then used to compute the Shannon information content. We then take the ceiling and add 2.

Using the prediction rule for the bigram model the maximum number of bits required to encode `thesis.txt` is 1,175,382.

Again the rule is computed for every character of the file and the probabilities are then used to compute the Shannon information content. Again, we take the ceiling and add 2.

2 Noisy Channel Coding

2.1 XOR-ing packets

The resulting string is `User: s5559183948`.

2.2 Decoding packets from a digital fountain

The algorithm takes as its input a list of packets and a list of sets containing identities of the source bytes that were used for each of the packets.

It loops through these lists zipped and only breaks out if all sets of identities are empty. Every time it encounters a set in position i with only one identity number k it pops k from the set and saves the character represented by the i-th packet in ASCII in the k-th position of the result list.

After this, k is removed from all the other sets that contain it. Furthermore, if any of these sets, say in position j, contains other numbers as well then the i-th packet is XOR'd with the packet in j-th position.

The source string is then re-constructed from the result list. The packets used to construct the string are also recorded and returned with the string.

The decoded string is `Password: X!3baA1z` and packets used are as follows: `16`, `23`, `2`, `21`, `22`, `20`, `8`, `10`, `17`, `19`, `6`, `7`, `9`, `11`, `14`, `15`, `12`, `13`.

2.3 Creating a code

I re-implemented the 2-dimensional parity-check code. It transforms a sequence of 4 source bits into a sequence of 8 bits, thus it is a (8, 4) block code. An n-dimensional parity-check code can correct $n/2$ errors. Thus this code can only correct bit sequences with 1 error.

2.3.1 Encoding

In the encoding step 4 parity check bits are added to the source sequence $x_1 \dots x_4$, computed as follows:

$$\begin{aligned} y_1 &= x_1 \oplus x_2 & z_1 &= x_1 \oplus x_3 \\ y_2 &= x_3 \oplus x_4 & z_2 &= x_2 \oplus x_4 \end{aligned} \tag{1}$$

The following message is then transmitted: $x_1\ x_2\ y_1\ x_3\ x_4\ y_2\ z_1\ z_2$.

2.3.2 Decoding

The transmitted message can be represented as:

x_1	x_2	y_1
x_3	x_4	y_2
z_1	z_2	

where every y_i represents the result of applying XOR to the rest of the bits in row i and every z_j represents the result of applying XOR to the rest of the bits in column j.

Upon receiving the transmitted message

$$x'_1\ x'_2\ y'_1\ x'_3\ x'_4\ y'_2\ z'_1\ z'_2$$

y_1, y_2, z_1 and z_2 are re-computed as in equations 1 and compared to the corresponding values received. If $y_i \neq y'_i$ for exactly one i and $z_j \neq z'_j$ for exactly one j then there was an error in the body of the message in row i and column j and we can correct it.

2.3.3 Example

For example consider source sequence 0100. We compute y_1, y_2, z_1 and z_2:

0	1	1
0	0	0
0	1	

and hence transmit message 01100001.

Let us say that when the decoder receives the message bit in row 2 and column 1 is flipped.

0	1	1
1	0	0
0	1	

Now, the decoder re-computes y_1, y_2, z_1 and z_4:

$$\begin{aligned} y_1 &= 1 = 1 = y_1' & \qquad z_1 &= 1 \neq 0 = z_1' \\ y_2 &= 1 \neq 0 = y_2' & \qquad z_2 &= 1 = 1 = z_2' \end{aligned} \tag{2}$$

and it can detect and correct the error as it now knows the error is in row 2 and column 1.

Therefore it produces the original message 0100.