Phil Morley

9060759686

Huffman Coding Project Questions

1) Explain the intuition underlying the construction of a Huffman code. How does it allow a document to be represented in fewer bits without losing any information?

A: The intuition is that words that are more likely to appear in a document encoded should be represented using less bits because they will most likely end up in the encoding more frequently. More of these smaller bit strings means a better compression. Where as the words that are less likely to appear in a document can be represented with longer bit strings because they will most likely not end up in the encoding.

2) Briefly explain how you calculated the compression ratio. What length did you calculate for the block code words?

A: To compute the compression, I first encoded the entire speech and measured the number of bits it used, will call this N. We’ll call the number of words that appeared in this document W and the number of unique words as U. I then computed the number of bits that would appear in a block encoding B = log2U . Finally, the compression = N / (W \* B).

3) What happens to the minimum, maximum, and average compression ratios when you increase the number of documents included in the set from which you build the code? Why?

A: The min, max, and average compression ratios all decrease as the number of documents included in the set decreases. The average is obviously decreasing because more of the documents cover more of the total vocabulary of words used in all of the speeches. The min and the max was a little surprising, I was actually expecting the opposite result. Although I can see that maybe this could be because the 2 most recent or the 2 oldest may have just been hard speeches to get good compression on. So increasing the number of documents would mean you’re more likely to cover a large portion of the vocabulary for some speech with short encodings of words.

4) Pick one of the codes that you produced. For that code, what is the length of the shortest code word? the longest? Does the number of documents used to construct the code affect these numbers?

A: I chose the code including all of the documents. The shortest code was 4 bits long 0010 encoding the word ‘of’. The longest code was 21 bits long 000100101011001001100 encoding the word ‘repaying’. As the number of documents represented in the code decrease from this, I would expect the shortest and longest code to increase. Since there would be less words with incredibly high probabilities showing up, there would be more nodes created from attaching other nodes higher in the tree making the shallowest leaf (shortest code) appearing lower in the tree. I would expect the longest code to stay about the same because there are still the same number of symbols encoded in the coding so the tree would tend to be around the same height. If anything it might get a little smaller because lots of small probability leaves would tend to make the tree fuller which would decrease the overall height.

5) For the code constructed from all of the speeches, find the ten speeches with the worst compression ratio and the ten speeches with the best compression ratio. Can you explain why their compression ratio is good or bad?

A: Most of the worst compression ratios came from speeches in the late 1700s and the early 1800s. It seems these speeches tend to use words that aren’t part of the vernacular of those speeches used post 1830. Presidents just talked differently then. So these different words were encoded with longer bit strings in the Huffman code. The best 10 compression ratios come from the 1830s-1860s as well as a 2009 speech. I found that these speeches tend to focus a lot on the talk of rights, war, legislature, etc.... These words tended to get repeated throughout the speech and they are also words that are generally discussed by presidents a lot. So these speeches had a lot of words that could be encoded with short strings.

6) What does it mean for Huffman codes to be prefix codes? How would you decode a document encoded using a Huffman code?

A: A prefix code means that as you’re reading an encoding, as soon as you’ve seen enough bits to represent some new character, than you stop looking, replace these bits with the new character’s bits. For example you could never have 1001 represent A and 10011 represent B. To decode, you’d walk down your code tree choosing a direction based on whether you see a 1 or 0. In general, 0 means left, 1 means right. As soon as you reach a leaf, you replace the bits you’ve read with the character at the leaf. This continues until the entire encoding is decoded.

7) If you worked as a team, what did each of the team members contribute?

A: I worked on this project alone.