Questions:

1) Explain the intuition underlying the construction of a Hu\_man code. How does it allow a document to

be represented in fewer bits without losing any information?

2) Briey explain how you calculated the compression ratio. What length did you calculate for the block

code words?

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?

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 a\_ect these numbers?

5) For the code constructed from all of the speeches, \_nd 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?

1

6) What does it mean for Hu\_man codes to be pre\_x codes? How would you decode a document encoded

using a Hu\_man code?

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

1)

2) The compression ratio was calculated by dividing the number of bits in a Huffman encoded file by the number of bits in that same file if it was block encoded. That is to say, after first creating and smoothing a dictionary we would read a file and add up the encoded length of every word in that file to get the total number of bits from Huffman encoding in a given file. Bloc encoding, on the other hand, was done by counting the number of words in a file and multiplying that number by the log base 2 of itself.

3) As you increase the number of documents included in the set the max, min, and averages all decrease. This is most obvious in the average of the entire ratio, especially when comparing the first trial (using only the oldest speech) versus all of the speeches. This is because we are using a larger sample base and thus can build the most efficient Huffman code.

4)

5)

6) Huffman coding is a type of prefix coding. That is to say that any bit-string that represents a word is never the prefix to another longer bit-string. Huffman encoded documents could be decoded by using the bits to traverse a tree whose leafs hold the value of whatever you are decoding.

7) Ethan wrote the majority of the code to do the actual encoding (i.e. The methods to traverse and build Huffman codes). Chelsey helped devise this code, via sudo-code, and also provided methods for data analysis and graphing.