Euan Rochester – Text processing assignment 2

My implementation is complete as specified in the assignment brief. Some effort has been made to optimize the size of the symbol-model file, such as splitting the HNode and HLeaf classes to prevent nil left and right being stored for leaves, and nil values being stored for Nodes.

It may also be possible to improve decoding speed by either tuning the size of the buffers for the InputBitstream, or foregoing the abstraction altogether. Decoding speed might also be improved by pre-computing a lookup-table of some sort rather than using the huffman tree directly for the decoding as binary trees have on average O(log n) speed, whereas for example, hashmaps have on average O(1) speed.

Symbol Model	Approximate Times In Seconds			File Sizes	
	Building Symbol Model	Encoding Input File	Decoding Compressed File	Compressed File	Symbol File
Characters	0.15	4.0	9.5	690362	2427
Words	0.95	1.9	5.0	392564	672153

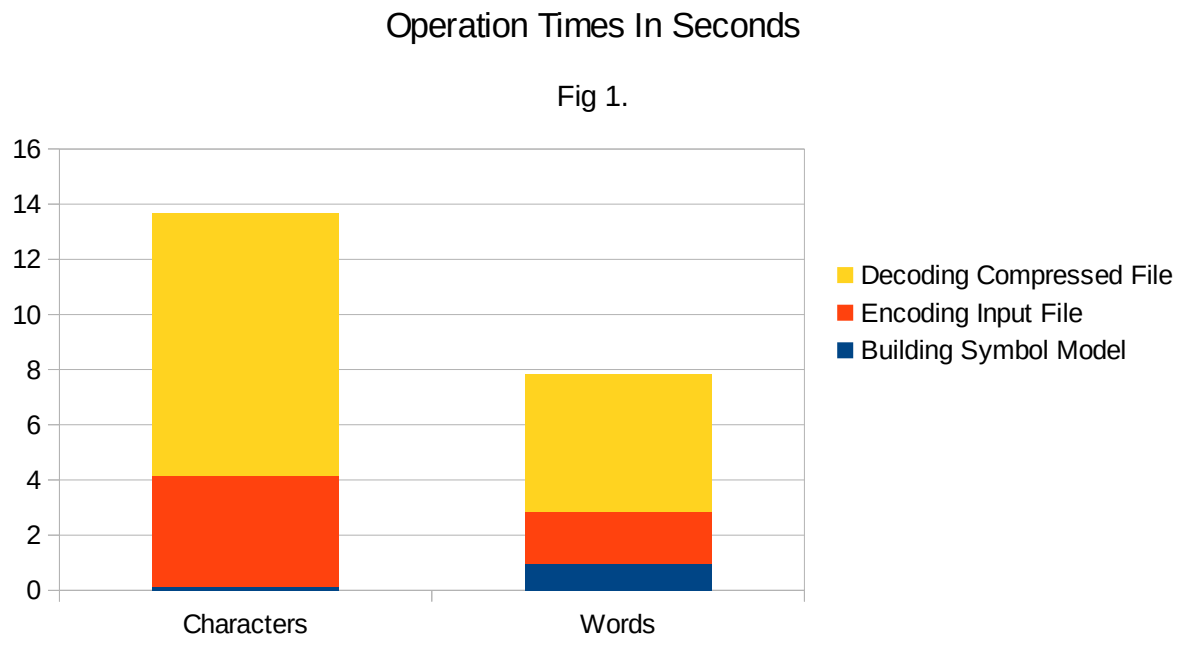

Fig 1.

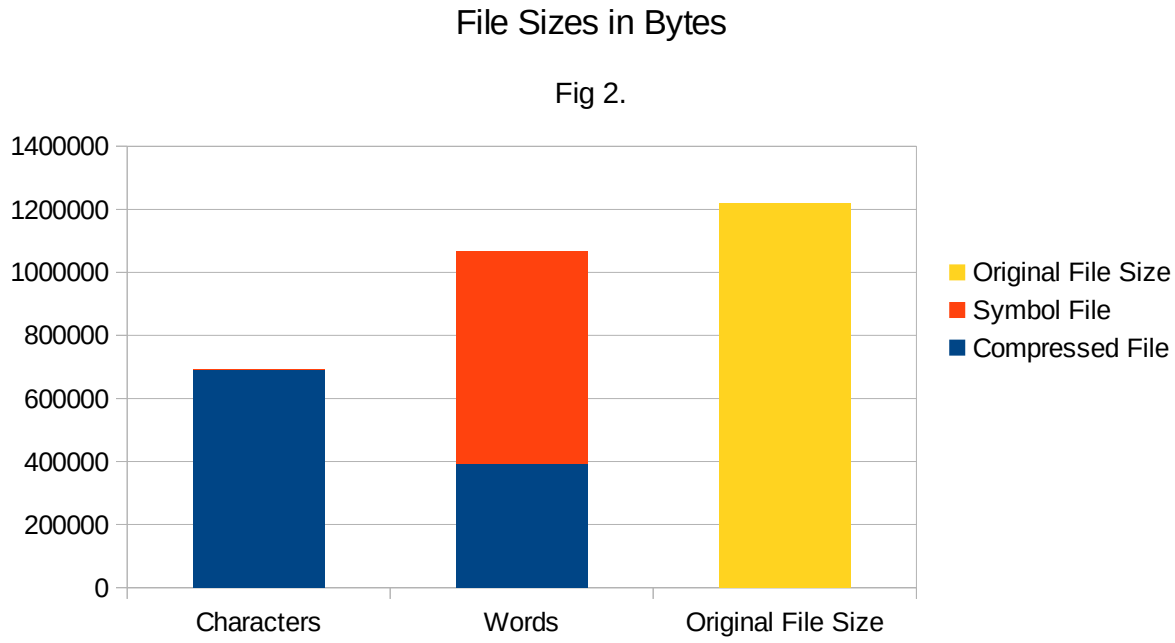

Fig 2.

In this case there is no clearly better model for all situations, as each has its own tradeoffs.

As can be seen from fig 2. one tradeoff between the word and character based models is that the compressed file is significantly smaller in the word based model, and the time to encode and decode are shorter, but the symbol model is significantly larger.

The word based compression would be most useful when the symbol model is pre-determined and not sent itself, or if the symbol model is used for a whole set of files, and the model is only sent once.

The character based compression is likely more useful for cases where the symbol model would be sent along with the file, such as in compressing a single file to be shared on the internet, as the overall size is smaller.