"Optimal" sounds pretty good!

Does that mean we can't do any better?

Well, not by encoding symbols one-at-a-time.

But if we want to encode long sequences of symbols, we can reduce the expected length of the encoding by working with, say, pairs of symbols instead of only single symbols.

The table below shows the probability of pairs of symbols from our example.

If we use Huffman's Algorithm to build the optimal variable-length code using these probabilities, it turns out the expected length when encoding pairs is 1.646 bits/symbol.

This is a small improvement on the 1.667 bits/symbols when encoding each symbol individually.

And we'd do even better if we encoded sequences of length 3, and so on.

Modern file compression algorithms use an adaptive algorithm to determine on-the-fly which sequences occur frequently and hence should have short encodings.

They work quite well when the data has many repeating sequences, for example, natural language data where some letter combinations or even whole words occur again and again.

Compression can achieve dramatic reductions from the original file size.

If you'd like to learn more, look up "LZW" on Wikipedia to read the Lempel-Ziv-Welch data compression algorithm.