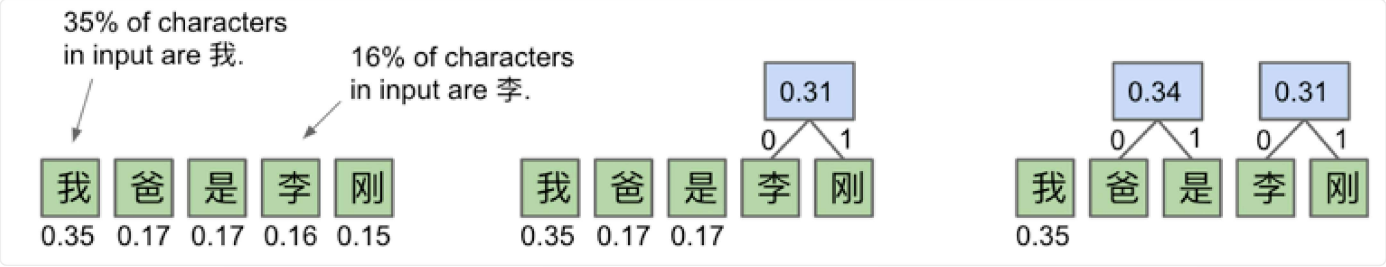
38.4 Huffman Coding Conceptuals

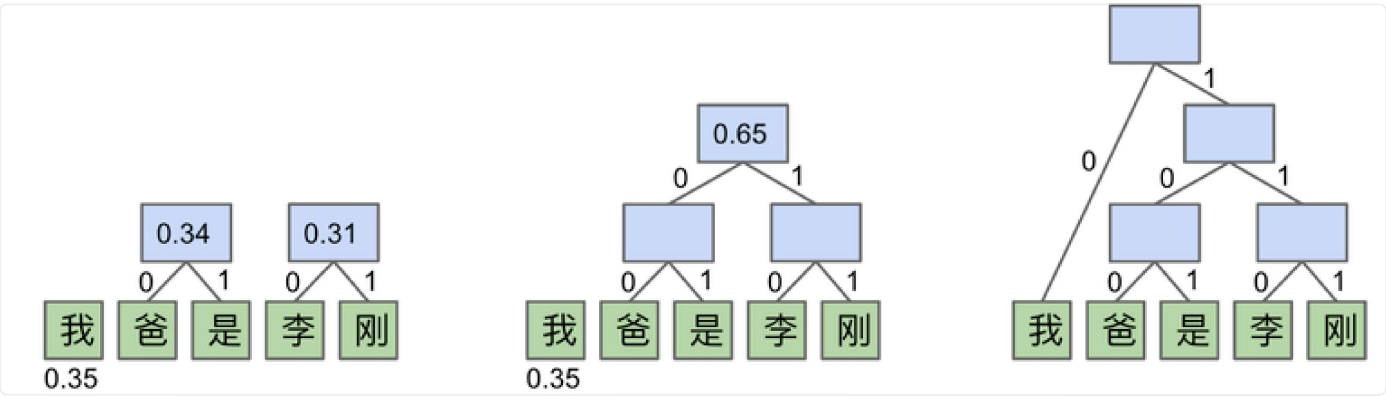
Core Idea

Huffman coding takes a bottom-up approach to prefix-free codes, as opposed to the top-down approach taken by Shannon-Fano codes. The algorithm is as follows:

- Calculate relative frequencies.
  - Assign each symbol to a node with weight = relative frequency.
  - Take the two smallest nodes and merge them into a super node with weight equal to sum of weights.
  - Repeat until everything is part of a tree.



Huffman Coding: step by step example (Part 1)



Huffman Coding: step by step example (Part 2)

Data Structures

Let's now think about the data structures we would use for the encoding and decoding processes of the Huffman Coding process. Recall that encoding will translate symbols to code words and decoding will do the opposite. An example is as follows:

- Encoding translates `I ATE` into `0000011000100101.`
- Decoding translates `0000011000100101` into `I ATE`.

> For encoding (bitstream to compressed bitstream), what data structure would we use?

> For decoding (compressed bitstream back to bitstream), what data structure would we use?

In Practice

Now that we have talked about what Huffman Coding is, let's talk about some practical issues that arise when we want to use it. There are two main philosophies we can use.

Corpus

- For each input type (English text, Chinese text, images), assemble huge numbers of sample inputs for each category. Use corpus to create a standard code for English, Chinese, etc.
- A corpus is a collection of pieces of language to be used as a sample of the language. Below is an example where we specify that we want to use the `ENGLISH` corpus to compress `mobydick.txt`

```
$ java HuffmanEncodePh1 ENGLISH mobydick.txt
```

Problem: Suboptimal encoding, which means our corpus does not exactly match our input. What this means in the context of this example is that `mobydick.txt` might not match up well with the general frequencies of `ENGLISH` texts and instead might have some other quirks or specifications from the author.

Unique Code

- For every possible input file, create a unique code just for that file. Then, when someone receives that file, they will know how to decode it using the code we send along the compressed file.
- As seen in the example below, we do not specify a corpus and we send along another file to help the decoding process for that specific file.

```
$ java HuffmanEncodePh2 mobydick.txt
```

Problem: This approach requires us to use extra space for the codeword table in the compressed bitstream. However, this generally works better than the corpus philosophy so is used commonly in the real world.

Previous
38.3 Shannon-Fano Codes

Next
38.5 Compression Theory

Last updated 1 year ago