Compression

Ulrik de Muelenaere

Topics:

- Information and Entropy
- Huffman Coding
- Arithmetic Coding

Compression

- Compression deals with encoding and decoding sequences of symbols.
- The symbols are taken from an alphabet

  $A = \{a_1, a_2, \ldots, a_n\}$.

- Any symbol from the alphabet can be represented using $log_2\, n$ bits.
- In the probability distribution $P = \{p_1, p_2, \ldots, p_n\}$, p_i gives the probability of the symbol a_i occurring.

Information

- We can define the information we gain by observing the occurrence of a symbol with probability p as $I(p) = -\log_2 p$.
- The base of the logarithm can be changed to use units other than bits.
- As an example, consider a fair coin toss. If the result is heads (or tails), we gain $I(0.5) = 1$ bit of information.
- If both sides of the coin are marked heads, we gain $I(1) = 0$ bits of information when the result is heads, since we already knew it would be heads.
- If a biased coin results in heads with probability 0.25, we gain $I(0.25) = 2$ bits of information if the result is heads, but only $I(0.75) \approx 0.42$ bits if it is tails.

Entropy

- Given a probability distribution P, we can calculate the average information gained per symbol using

  $$H(P) = \Sigma_{p \in P}\, p\, I(p)$$

  $$= - \Sigma_{p \in P}\, p \log_2 p$$

- This is known as the entropy, which is also a measure of uncertainty.
- Shannon's source coding theorem states that the entropy places a lower bound on compressed size. A sequence of n symbols, with probability distribution P, cannot be compressed (without loss of data) into less than $n\, H(P)$ bits (as n tends to infinity).

Entropy

- It can be shown that

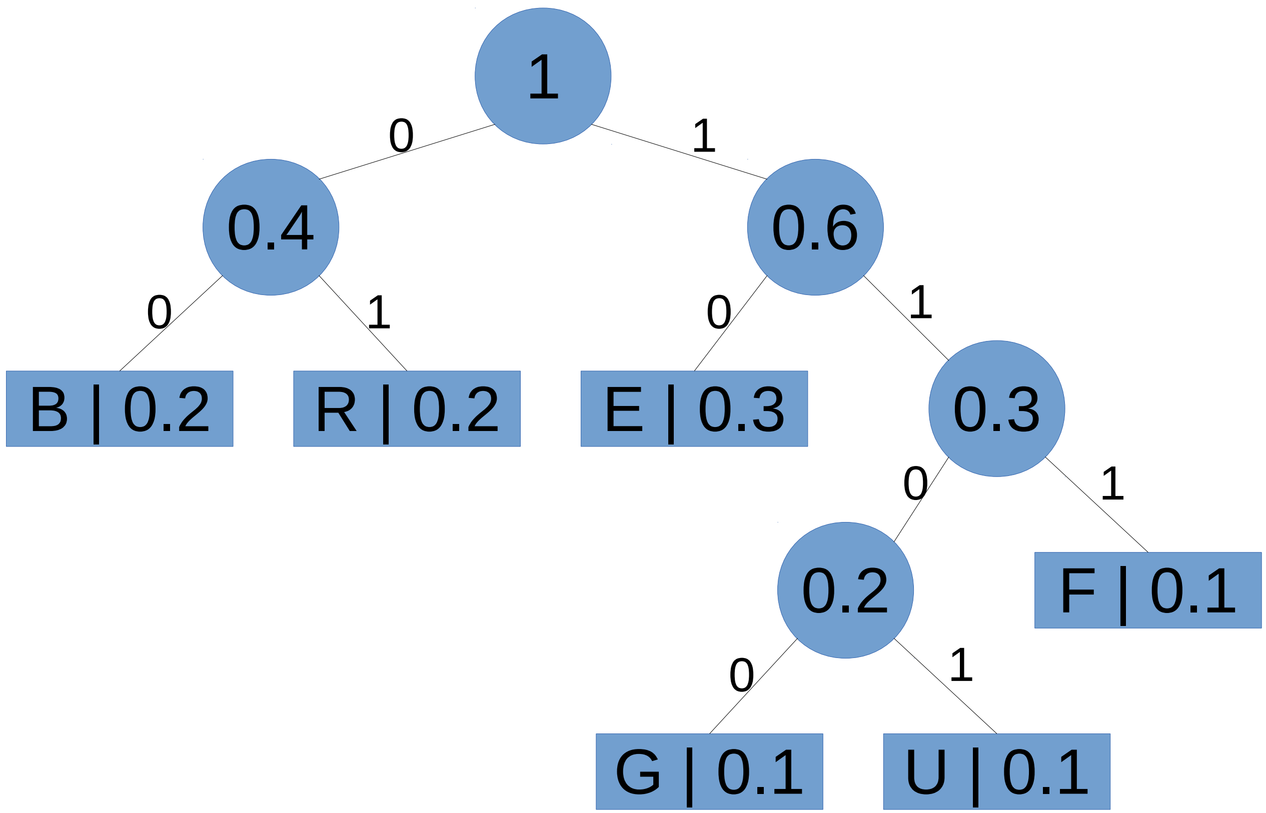
  $0 \leq H(P) \leq log_2\ n$.

- $H(P) = 0$ when a single symbol has $p = 1$ and all other symbols have $p = 0$.
- $H(P) = log_2\ n$ when all symbols have equal $p = 1 / n$.
- This shows that evenly distributed probabilities produce a high entropy, which limits compression. Uneven distributions lower entropy, allowing for more compression.

Huffman Coding

- Huffman coding encodes each symbol as a string of bits. Symbols that have a higher probability are assigned shorter bit strings than those that have a lower probability.
- Bit strings are assigned to symbols using a prefix code. This means that the bit string representing a symbol is never a prefix of the bit string representing another symbol.
- This property allows for easy decoding. Once the decoder recognizes a bit string, it can decode the symbol, without trying to determine if it is a prefix of a longer bit string.
- A prefix code can be represented as a binary tree. A 0 in the bit string means following the left child and a 1 means following the right child. The symbols are the leaves of the tree and each internal node has 2 children.

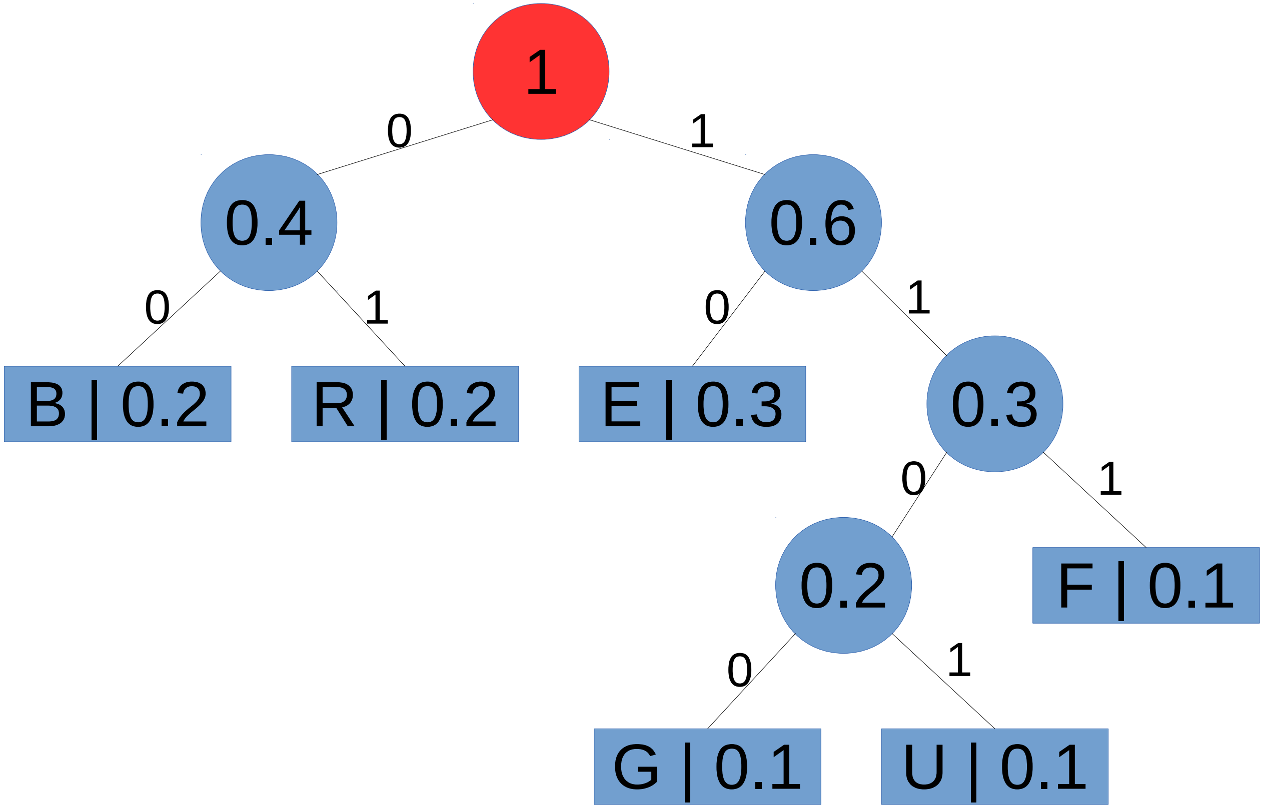
Huffman Coding



BEEFBURGER

0010101110011010111001001

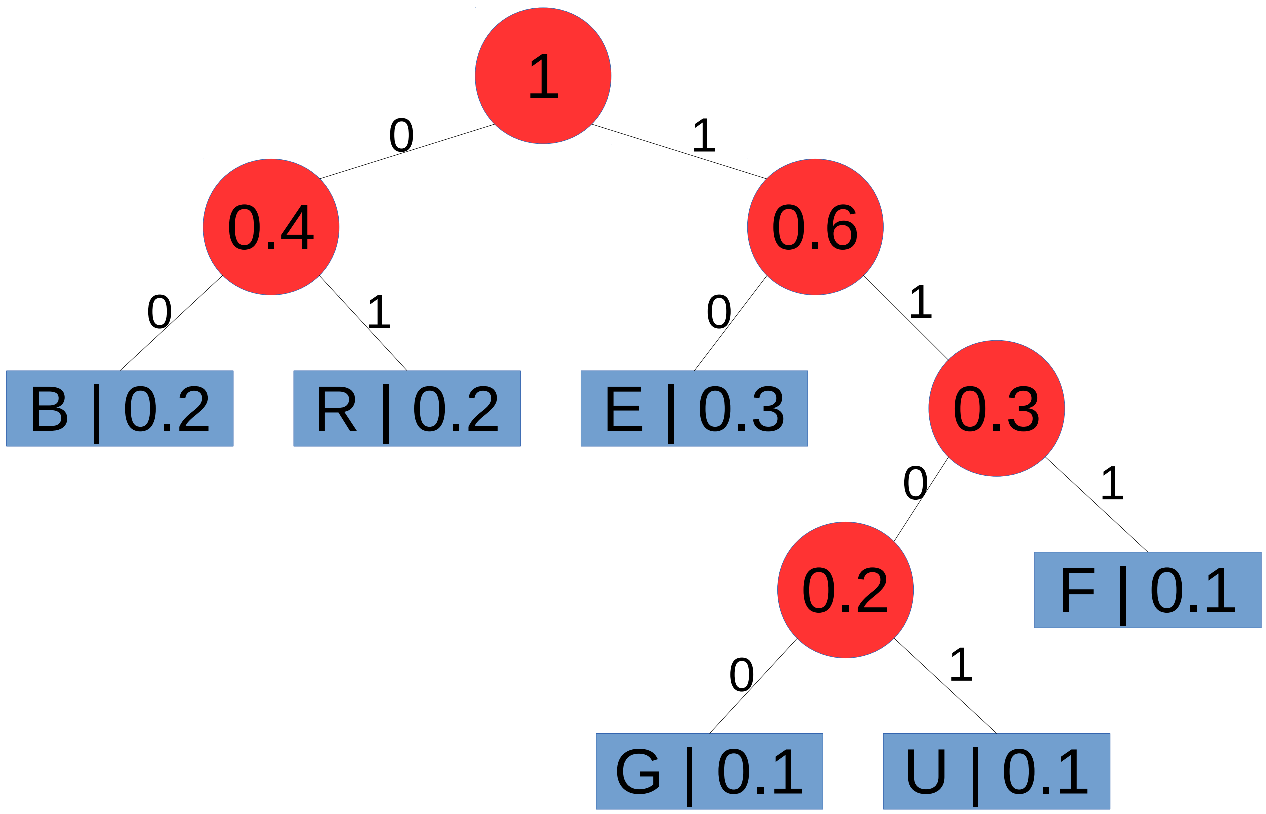
Huffman Coding



BEEFBURGER

0010101110011010111001001

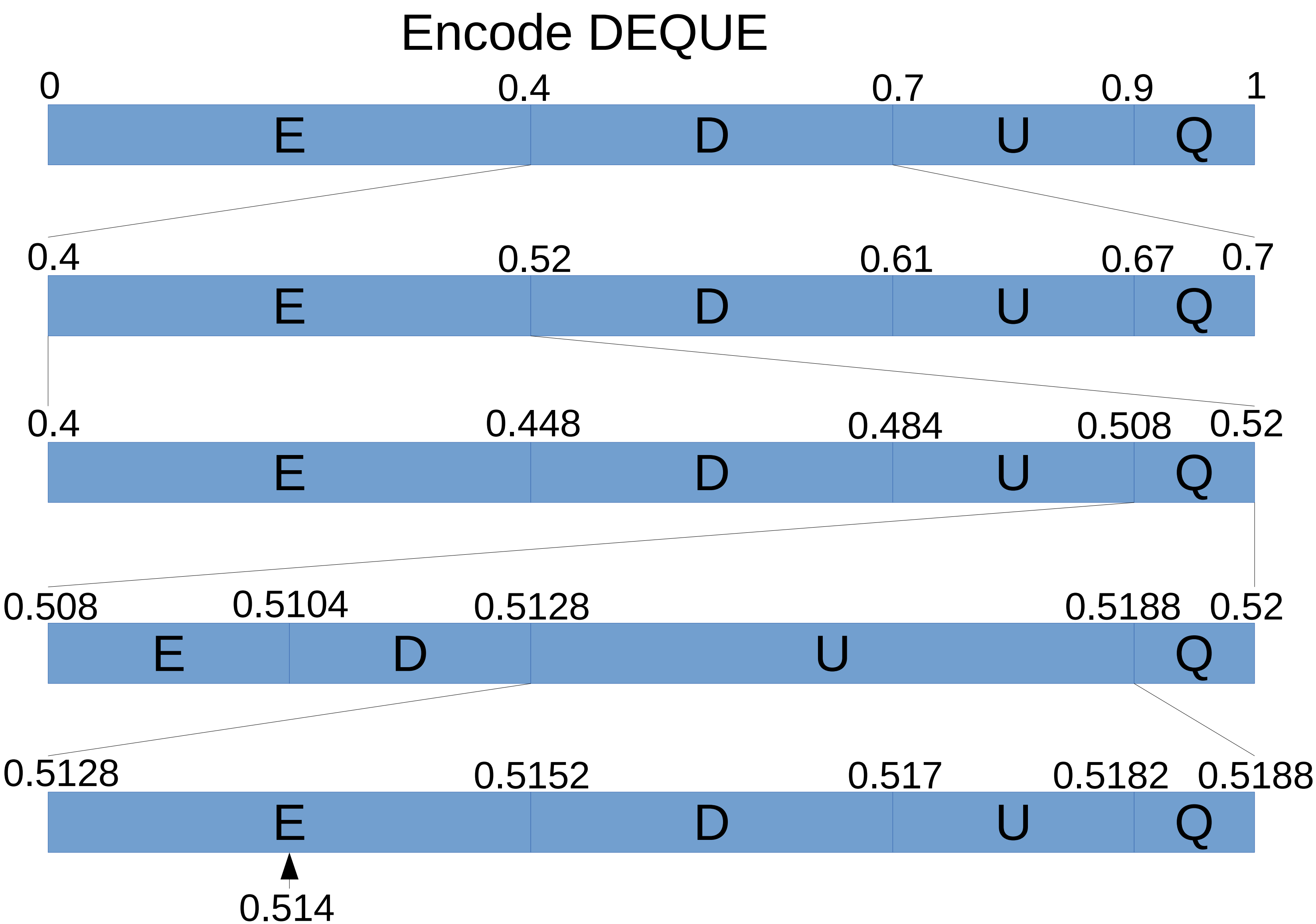
Huffman Coding



Arithmetic Coding

- Arithmetic coding encodes the entire message into a single fraction n, $0 \le n < 1$.
- This allows it to come closer than other algorithms to the optimal encoding specified by the entropy, in many situations.
- The probability distribution can change as each symbol is encoded. This also allows for a more optimal encoding, as long as the encoder and decoder change the probabilities in the same way. This can be used to create coders optimized for a specific language.
- This is only an introduction. We will not discuss implementation details, such as achieving the necessary precision.

Arithmetic Coding



Arithmetic Coding