CS3302 Practical 2: Hamming Codes

150023118

November 24, 2017

Overview

The objective of this practical was to implement Hamming codes and allow the user to experiment with Hamming codes through a GUI. The user was to be able to specify a number from 2 to 6 for parameter r and encode words with the corresponding Hamming encoding. Additionally, the application corrupts codewords at a certain probability, which the parity-checker tries to correct. The user is able to experiment with different error rates and parameters.

Usage

This application requires Python 3 (tested on version 3.6.3). Additional dependencies can be installed using

```
$ pip3 install -r requirements.txt
```

After all dependencies have been set up, the application can be launched with:

```
$ cd hamming_app/
$ python3 flask_app.py
```

Once the application is running, the main window should be accessible on any browser at http://127.0.0.1:8080.

Design

This application was written as a Python web application that delivered though Flask. The code for serving the pages is contained within `flask_app.py`, and the classes for the Hamming encoder and checker are included in `hammingclasses.py`.

Instances of the class `HammingEncoder` takes in a word and returns a codeword with the original bits in the word with parity bits attached. The constructor takes in the parameter r as its argument and generates a Hamming encoder with a generator matrix.

The generator matrix is configured so that the parity bits are as follows: if b_i stands for the bit at position i (that is, index $i - 1$), the parity bits are

$$P = \{\, b_i \mid \exists\, j \in \mathbb{N} \text{ such that } 2^j = i \;\&\; i < n \,\}$$

where P is the set of all parity bits. The bit is the xor of all bits for which a bitwise AND is nonzero, that is,

$$\text{for all } b_i \in P = \{\, b_i \mid \exists\, j \in \mathbb{N} \text{ such that } 2^j = i \;\&\; i < n \,\}$$

Testing