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 the Hamming encoder and checker are included in `hammingclasses.py`, which the user can interact with through the pages served by `flask_app.py`

Hamming encoder and checker

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 to place the parity-check bits at positions which are powers of 2. In other words, if b_i stands for the bit at position i (index $i-1$) and the set of all parity-bits is P,

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

Bit b_i is an even parity-check bit on every data bit b_j where $b_i \odot b_j \neq 0$; that is,

$$b_i = \bigoplus_j b_j \text{ for } \{\, b_j \mid b_i \odot b_j \neq 0\}$$

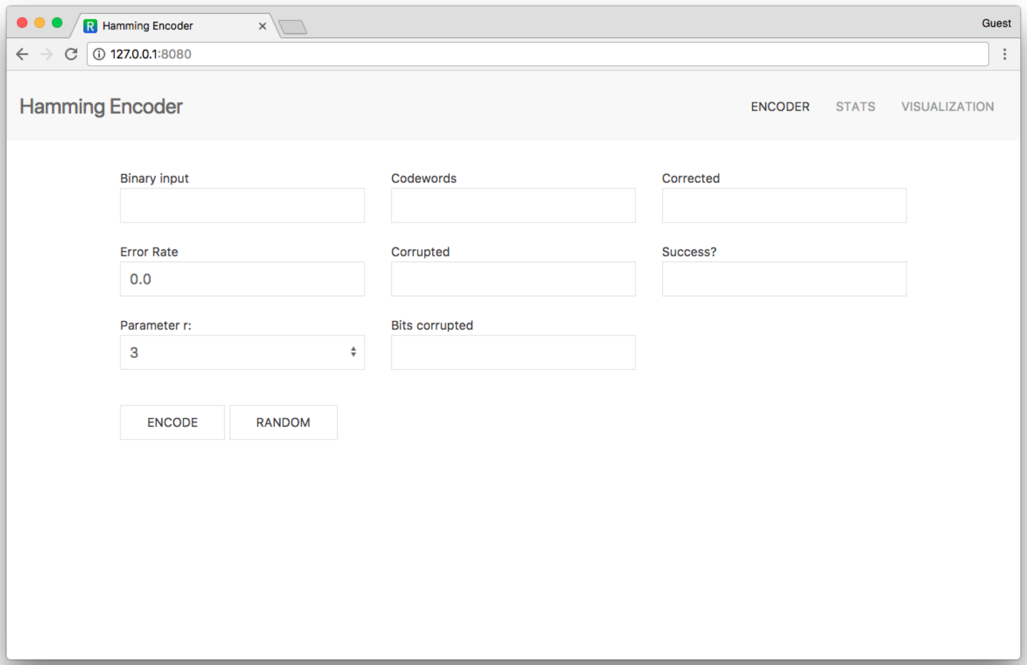
With this matrix, constructing codewords for each word becomes a matter of simply multiplying the word by the generator matrix. Since every number between 1 and n is uniquely represented in binary, it is also uniquely represented by a set of parity-check bits. Thus, this method allows us to construct generator matrix for Hamming codes of any parameter r where any one corrupted bit would be identified.

The class `HammingChecker` also takes in r as the parameter, and constructs a checker matrix using the same algorithm. Codewords are checked by multiplication with the checker matrix. If the resulting vector matches up with any row in the checker matrix, the bit of that row has been corrupted. If the resulting vector is a zero vector, no corruption has taken place.

User interface

Hamming encoding page

When the application is launched, the user should be greeted with a page that looks like the following.



Binary input

Codewords

Corrected

Error Rate: 0.0

Corrupted

Success?

Parameter r: 3

Bits corrupted

Testing