Homework Assignment #2

Channel Encoding: Error Correction

By

Zuoda Ren

Zhening Li

Zeqi Chen

Weimin Zhou

Faculty Advisor:

Dr. Paul S. Min (Electrical and System Engineering)

ESE471

Department of Electrical and System Engineering

Washington University

Saint Louis, Missouri

Mar 6, 2016

Methods

In this assignment, our group tackled the channel encoder. As professor have explained during the lecture, the channel encoder converts the output from the source encoder to match the requirements imposed by the channel, such as to provide reliable communication in the presence of errors. Errors are unavoidable in all realizable channels, no exception. So, we write an m file to simulate the real channel.

Our code can be roughly divided into five parts.

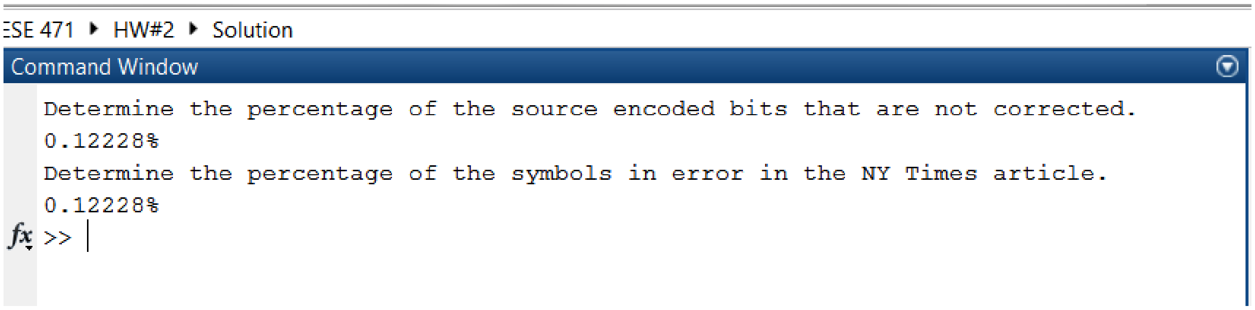
1. Copy the article from Word into a TXT file which can be processed more conveniently in Matlab, then using "fopen" to open the original file.
2. Mimic the channel encoder by storing the article's Huffman encode in an article-length cell corresponding to every word's ASC-II, then triplicate every symbol's Huffman code.
3. To be more realistic, we use *rand* function to determine whether each bit is in error.
4. Then, apply the majority reasoning to correct the errors. In Matlab, *mode* function can realize it.
5. Compare the input file's Huffman code and the output file's Huffman code, also, the input article and output article to get the percentage of errors, then show them on the terminal window.

For more details, we attached our Matlab code in Attachment#1 and add many comments in it. Please check it!

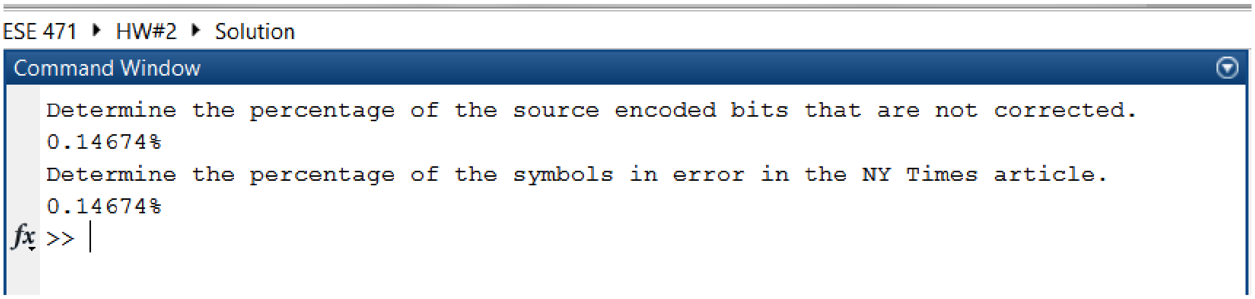
Results

For different error probabilities p, we simulated this channel encoder three times and got different results.

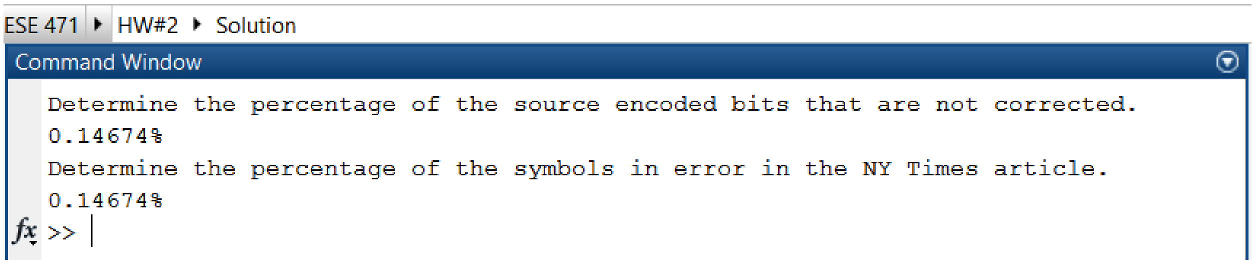
When p = 0.01:



```
Determine the percentage of the source encoded bits that are not corrected.
0.12228%
Determine the percentage of the symbols in error in the NY Times article.
0.12228%
fx >> 
```

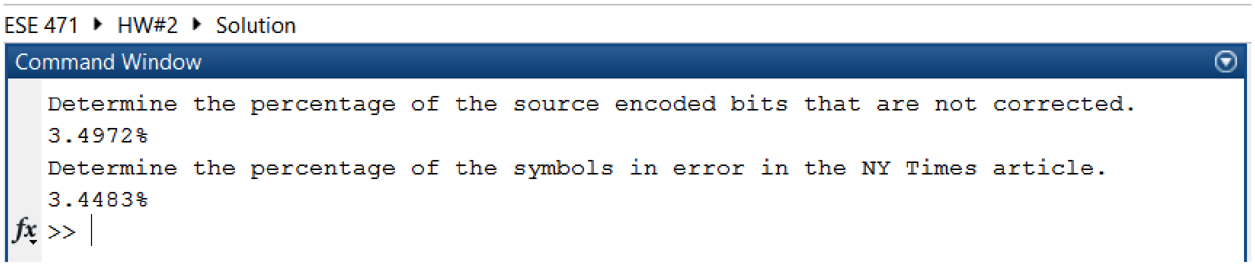


```
Determine the percentage of the source encoded bits that are not corrected.
0.14674%
Determine the percentage of the symbols in error in the NY Times article.
0.14674%
fx >>
```

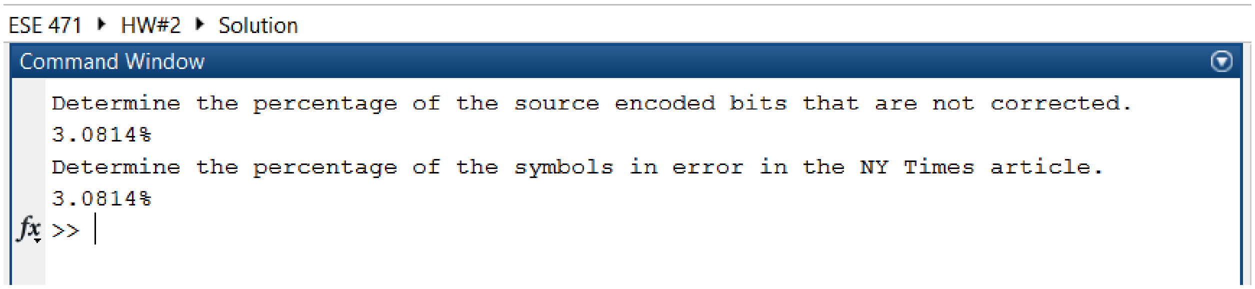


```
Determine the percentage of the source encoded bits that are not corrected.
0.14674%
Determine the percentage of the symbols in error in the NY Times article.
0.14674%
fx >>
```

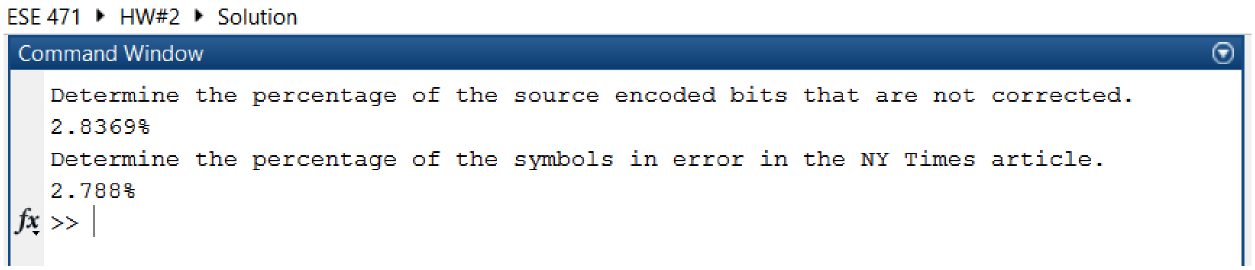
When p = 0.05:



```
Determine the percentage of the source encoded bits that are not corrected.
3.4972%
Determine the percentage of the symbols in error in the NY Times article.
3.4483%
fx >>
```

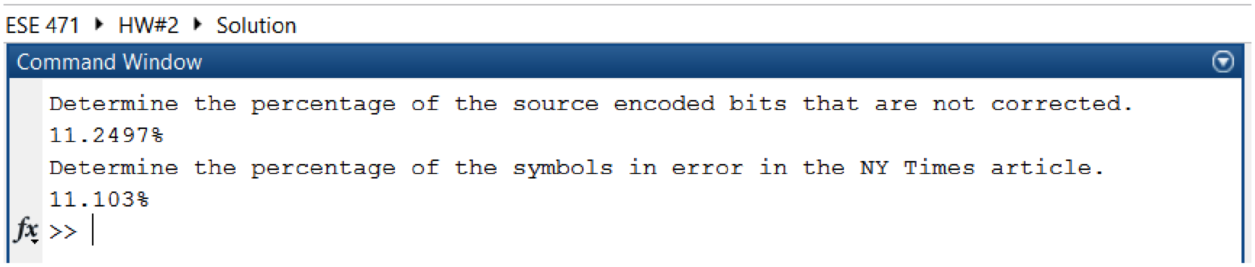


```
Determine the percentage of the source encoded bits that are not corrected.
3.0814%
Determine the percentage of the symbols in error in the NY Times article.
3.0814%
fx >>
```

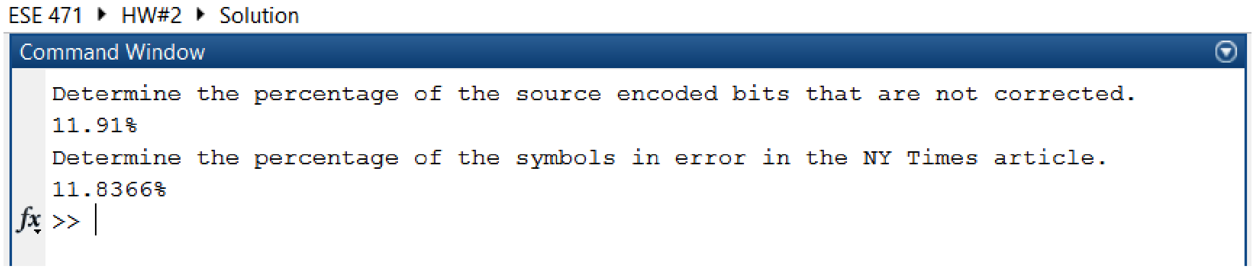


```
Determine the percentage of the source encoded bits that are not corrected.
2.8369%
Determine the percentage of the symbols in error in the NY Times article.
2.788%
fx >>
```

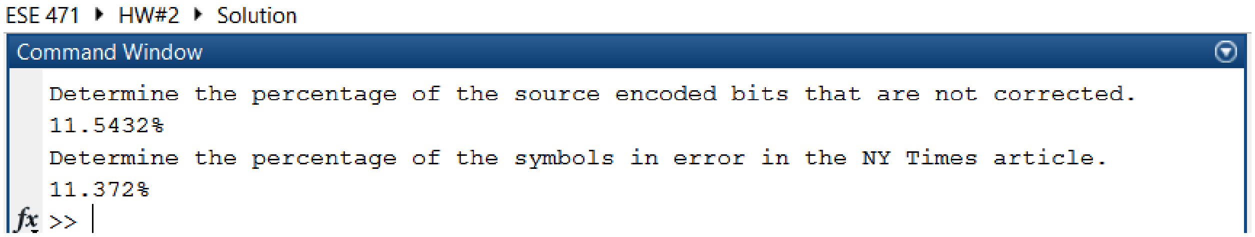
When p = 0.1:



```
Determine the percentage of the source encoded bits that are not corrected.
11.2497%
Determine the percentage of the symbols in error in the NY Times article.
11.103%
fx >> |
```



```
Determine the percentage of the source encoded bits that are not corrected.
11.91%
Determine the percentage of the symbols in error in the NY Times article.
11.8366%
fx >> |
```



```
Determine the percentage of the source encoded bits that are not corrected.
11.5432%
Determine the percentage of the symbols in error in the NY Times article.
11.372%
fx >> |
```

Notice that the increase of artificially added error probability will led to the increase of the percentages of both the source encoded bits that are not corrected and the symbols in error in the NY Times article. When each channel encoded bit becomes an error with probability p = 0.1, the article error rise nearly 11% which is very high error rate. Also, when the error probability is every small, the percentage of source encoded error and the percentage of symbols error are nearly the same.

Attachment #1 Implement in Matlab

```
%%
%Author: Zhening Li zhening.li@wustl.edu
%
%Purpose: Simulate channel encoder in different error probabilities
%
clc
clear
%%
%Open file
file_in  = fopen('NY_time.txt','r','n','UTF-8');%open the original file
file_out = fopen('NY_time_binary.txt','w');%create a file for output
file_original = fread(file_in,Inf,'char','n');%read the article from the
input file ASCII encode form

%%
%
%Define the matrices and cells
%
p = 0.1;%set the error probability p = 0.01 or 0.05 or 0.1
symbol_asc2 = [48;49;50;51;52;55;57;32;45;44;
               59;58;33;63;46;39;34;40;41;97;
               65;98;66;99;67;100;68;101;69;102;
               70;103;71;104;72;105;73;106;74;107;
               75;108;76;109;77;110;78;111;79;112;
               80;113;114;82;115;83;116;84;117;118;
               86;119;87;120;121;122];%store the ASCII encode of symbols

symbols_huffman = Huffman_encoding(symbol_asc2);%get the Huffman dictionary
[symbols_number,~] = size(symbol_asc2);%get number of the symbols
[article_length,~] = size(file_original);%get the length of the article
%define matrices and cells to store different codes
filein_binary = cell(article_length,1); %input file Huffman encode - cell
filein_binary_triple = cell(article_length,1); %triplicated encode - cell
fileout_binary = cell(article_length,1); %output file Huffman encode - cell
fileout_asc2 = zeros(article_length,1); %output file ASCII encode - matrix
error_huffman = zeros(article_length,1);%store the compared error - matrix
%%
%channel encoder
%Get the triplicated code of this article
%the forward error correction

for i = 1:article_length%go through the entire article
    symbol_current = file_original(i,1);%get the ASCII of the current symbol
    %find its corresponding index in huffman dictionary
    file_original_index = find(symbol_asc2 == symbol_current);
    %get its huffman code according to above index
    file_binary = symbols_huffman(file_original_index,2);
    filein_binary(i,1) = file_binary;%store the original article Huffman
encode
    %triplicate its huffman code
    file_binary = cell2mat(file_binary); %transfer this cell to matrix form
    filein_binary_repmat = repmat(file_binary,3,1);%triplicate
    filein_binary_repmat = filein_binary_repmat(:);
    filein_binary_repmat = filein_binary_repmat';
    filein_binary_repmat =
mat2cell(filein_binary_repmat,1,length(filein_binary_repmat));
    %store its triplicated code
    filein_binary_triple(i,1) = filein_binary_repmat;
end
%%
%
%Add channel errors
%
for i = 1:article_length%go through the entire article
    filein_binary_current = filein_binary_triple(i,1);
    filein_binary_current = cell2mat(filein_binary_current);

    uniform_error = rand(1,length(filein_binary_current));%use rand function
    error_index = find(uniform_error<=p);%determine add error to which index
    filein_binary_current(:,error_index) =
~filein_binary_current(:,error_index);%add error
    filein_binary_current =
mat2cell(filein_binary_current,1,length(filein_binary_current));
    filein_binary_triple(i,1) = filein_binary_current;%store back this
triplicated code with error
end
%%
%receiver
%decode the triplicated encode using the majority reason
%
for i = 1:article_length
    triple_current = filein_binary_triple(i,1);
    triple_current = cell2mat(triple_current);
    triple_current_1 = triple_current(:,1:3:end);%create a new matrix by
picking elements every three elements
    triple_current_2 = triple_current(:,2:3:end);
    triple_current_3 = triple_current(:,3:3:end);
    triple = [triple_current_1;triple_current_2;triple_current_3];
    fileout_binary_current = mode(triple);%use mode function to apply the
majority reason
    fileout_binary_current =
mat2cell(fileout_binary_current,1,length(fileout_binary_current));
    fileout_binary(i,1) = fileout_binary_current;
end
%%
%transfer huffman encode to corresponding ASCII
%
for i = 1:article_length
    fileout_binary_current = fileout_binary(i,1);
    for j = 1:symbols_number
        huffman_code_check = symbols_huffman(j,2);
        flag_equal = isequal(fileout_binary_current,huffman_code_check);
        if(flag_equal==1)
            symbol_out_asc = symbol_asc2(j,1);
            break
        end
    end
    fileout_asc2(i,1) = symbol_out_asc;
end
%%
%write the output file
```

```
fwrite(file_out,fileout_asc2,'char');
%close all files
fclose(file_in);
fclose(file_out);
%%
%calculate  the percentage of errors
%
%Determine the percentage of the source encoded bits that are not corrected
for i = 1:article_length
error_huffman(i,1) = isequal(filein_binary(i,1),fileout_binary(i,1));
end
percentage_error_huffman =
length(find(error_huffman==0))/length(error_huffman);
%Determine the percentage of the symbols in error in the NY Times article.
error_article = file_original==fileout_asc2;
percentage_error_article =
length(find(error_article==0))/length(error_article);
%%
%display the percentage of errors in the terminal window
disp_error_1 = [num2str(percentage_error_huffman*100),'%'];
disp('Determine the percentage of the source encoded bits that are not
corrected.');
disp(disp_error_1);

disp_error_2 = [num2str(percentage_error_article*100),'%'];
disp('Determine the percentage of the symbols in error in the NY Times
article.');
disp(disp_error_2);
```

Attachment #2 Function to calculate the Huffman encode

```
function dict = Huffman_encoding(symbols_input)
%function dict = Huffman_encoding()
%
%calculate the Huffman encode
%
%symbols_input --- the ASCII matrix of symbols which need to be encoded
%
symbols = symbols_input; % Distinct symbols that data source can produce
p = [];% Probability distribution

[dict,~] = huffmandict(symbols,p); % Create dictionary.
```