UNIVERSIDADE
DE ÉVORA

Relatório
Teoria de Informação

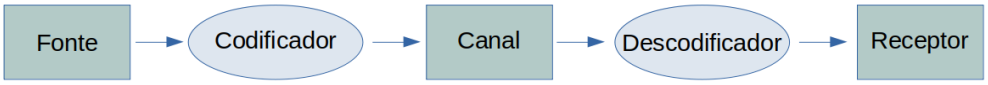
Rúben Peixoto, nº 37514

Janeiro 2020

Contents

1 Introdução

Neste trabalho foi implementado um sistema de transmissão de dados, tal como demonstra a figura a seguir.



Neste esquema tanto a fonte como o canal são dois programa fornecidos, o codificador e descodificador são dois programas implementados por mim e, por fim, o receptor serão três ficheiros em que o primeiro contém o input com redundância, o segundo contém o input comprimido com o algoritmo Lempel-Ziv-Welch (LZW) e o terceiro ficheiro com o input da fonte.

Neste contexto, acrescento, que no programa "Codificador" é aplicado redundância aplicando o algoritmo de Hamming e no programa "Descodificador" são aplicados os algoritmos para detectar e corrigir possíveis erros que possam ter surgido do canal. É neste também que é aplicado o algoritmo de compressão e descompressão. O algoritmo de descompressão só foi implementado para verificar se é possível recuperar a mensagem inicial.

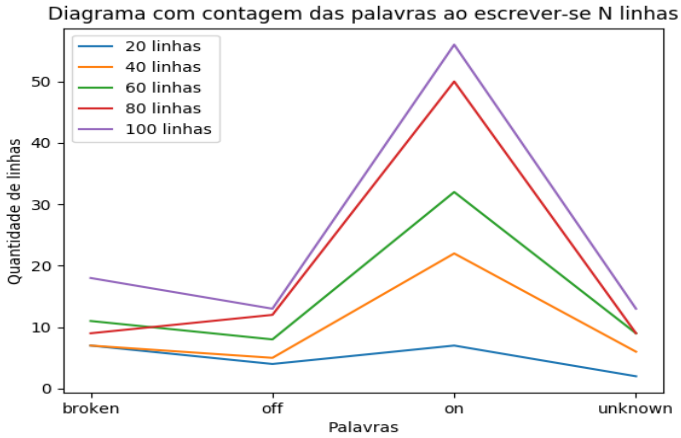
A linguagem utilizada para este trabalho foi Python com a versão 3.7.

2 Modelo Probabilístico

Neste tópico procuro estudar o funcionamento tanto da fonte como do canal. Para a fonte achei interessante verificar qual é a palavra mais provável e a menos provável de aparecer e no canal, qual a posição que tem mais propensão a erros.

2.1 Fonte

Neste estudo preocupei-me em ver quais as palavras mais prováveis de aparecer e as menos prováveis.



Para verificar quais as palavras que tinham mais e menos frequência, criei um programa que recebe N linhas da fonte e calcula a quantidade de vezes que estas apareceram. O valor "N" assume os valores 20, 40, 60, 80 e 100. Tal como aparece no gráfico a baixo.

Para 100 linhas geradas pela fonte pode-se verificar que as probabilidades das palavras, "on", "off", "broken" e "unknown" são:

$$P(on) = \frac{56}{100};\ P(off) = \frac{13}{100};\ P(broken) = \frac{18}{100};\ P(unknown) = \frac{13}{100}$$

Com isto concluí-se que a surpresa de aparecer as palavras "off" ou "unknown":

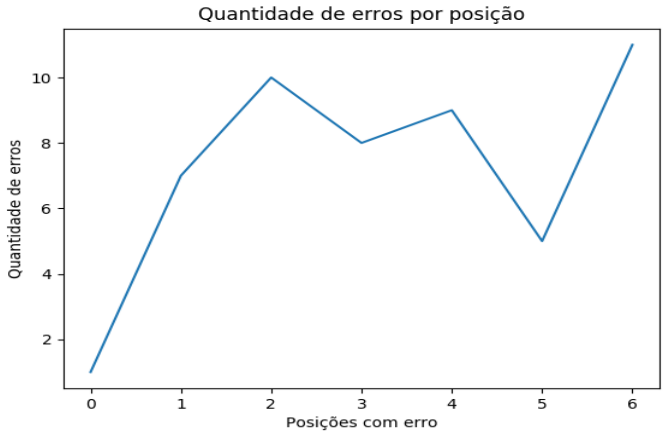
$$S(off) = -log_2(\frac{13}{100}) \simeq 2.94\ bits$$

é maior que a surpresa da palavra "on", tal como seria esperado.

$$S(on) = -log_2(\frac{56}{100}) \simeq 0.84\ bits$$

2.2 Canal

Para estudar o canal, criei um programa que verifica as posições onde surgiram mais erros numa linha. Este teste foi realizado para 200 linhas.



Neste gráfico pode-se verificar que a 1ª posição da linha tem uma probabilidade muito baixa de ocorrer um erro, neste caso, foi de 1 erro por 200 linhas e a posição que tem maior probabilidade é a ultima.

Uma outra situação que pude observar, é que, dos bits que compõem um char apenas a ultima posição é que é alterada.

Ou seja, para corromper um char é realizado um XOR entre esse char e o número 1_2.

3 Compressão

Para este trabalho o algoritmo de compressão utilizado foi o Lempel-Ziv-Welch (LZW).

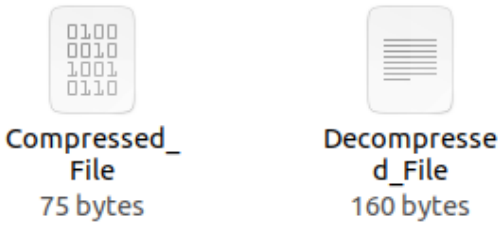


Figure 1: Neste exemplo foram comprimidas 20 palavras utilizando o LZW.

O retorno deste algarismo é uma lista em que os primeiros X símbolos são todas as palavras diferentes encontradas pelo o algoritmo. Os seguintes símbolos são os índices do dicionário colocados no output. Para separar uma informação da outra foi colocado um símbolo nulo entre estas.

4 Modelo do canal e Capacidade

Na minha opinião o canal utilizado neste trabalho é a chamada "Máquina de escrever ruidosa", isto porque, embora o alfabeto da fonte seja ("o", "n", "f","u","k","w","\n","-"), é possível receber e devolver outros símbolos que não constam neste conjunto.

Como o canal tem 127 símbolos de entrada e 127 símbolos de saída pode-se dizer que a capacidade deste canal é:

$$C = \max_{p(x)} I(X,Y) = \log_2 63.5 \simeq 1.80 \text{ bits}$$

5 Código de Hamming

Tal como falado na introdução, apliquei redundância na informação antes de este passar pelo canal. Para tal, o número de bits de dados foi de 7. Com isto, a palavra de código tem tamanho 11 bits e a redundância 4 bits.

Escolhi este tamanho porque queria trabalhar com tamanho de um char, que neste caso, tem 7 bits de informação.

Se passarmos 20 palavras pelo canal a probabilidade de um char aparecer trocado é de:

$$P(trocado) = \frac{7}{160} \simeq 4{,}38\%$$

Em que 160 é o número de chars das 20 palavras e 7 é o número de chars trocados. Com isto a probabilidade de não ocorrer erro é:

$$P(\overline{error}) = (\frac{153}{160})^{11} + 11(\frac{153}{160})^{10} \frac{7}{160} \simeq 30{,}77\%$$

6 Conclusão

Dos algoritmos criados neste trabalho, o único que me surgiu dúvida na sua implementação foi o código Hamming. Pois existe casos em que o output sai correcto e outros errado.

O resto do programa está a funcionar como desejado.

7 Webgrafia

- Slides do professor
- https://www.youtube.com/watch?v=m2dFjkZVlVw