Teoria da Informação

Trabalho 2015/2016

UNIVERSIDADE
DE ÉVORA

Realizado por:
João Calhau - 31621
José Pimenta - 31677

Introdução:

Este trabalho enquadra-se na disciplina de Teoria de Informação, e sendo um trabalho final tem como finalidade a aplicação dos conhecimentos adquiridos ao longo desta disciplina. Portanto neste trabalho vamos abordar temas como probabilidades, características de fonte, capacidade de canal, codificação, compressão, entre outras.
Mais especificamente iremos pegar num ficheiro com letras A,C,T,G que codificam o ADN e iremos utilizar técnicas de codificação e compressão de modo a enviar um excerto de ADN por um canal fictício que sofreria erros, sendo depois descodificado para o estado original, corrigindo os erros que se verifiquem possíveis.
Iremos assim tentar dar o nosso melhor, e iremos tentar utilizar os métodos que achemos mais corretos ou propícios à boa evolução do trabalho e que no final se concretize o que nos é pedido.

Tarefa 1:

Na tarefa 1 somos incumbidos de estudar as características da fonte. Primeiro fizemos um teste de letra a letra para ver em que percentagem aparecia cada letra:

Tamanho total da palavra: 11008

Letra A: 2374 vezes, ou seja em 21.57 % da palavra
Letra C: 2874 vezes, ou seja em 26.11 % da palavra
Letra G: 2672 vezes, ou seja em 24.27 % da palavra
Letra T: 3088 vezes, ou seja em 28.05 % da palavra

Entropia = 1.993 bits

De seguida fizemos um teste de combinações 2 a 2:

Tamanho da palavra: 5504 (metade da palavra original)

Combinação AA: 238 vezes, ou seja em 4.32 % da palavra
Combinação AC: 244 vezes, ou seja em 4.43 % da palavra
Combinação AG: 460 vezes, ou seja em 8.36 % da palavra
Combinação AT: 238 vezes, ou seja em 4.32 % da palavra

Combinação CA: 393 vezes, ou seja em 7.14 % da palavra
Combinação CC: 491 vezes, ou seja em 8.92 % da palavra
Combinação CG: 95 vezes, ou seja em 1.73 % da palavra
Combinação CT: 488 vezes, ou seja em 8.87 % da palavra

Combinação GA: 384 vezes, ou seja em 6.98 % da palavra
Combinação GC: 337 vezes, ou seja em 6.12 % da palavra
Combinação GG: 537 vezes, ou seja em 9.76 % da palavra
Combinação GT: 308 vezes, ou seja em 5.60 % da palavra

Combinação TA: 179 vezes, ou seja em 3.25 % da palavra
Combinação TC: 335 vezes, ou seja em 6.09 % da palavra
Combinação TG: 430 vezes, ou seja em 7.81 % da palavra
Combinação TT: 347 vezes, ou seja em 6.30 % da palavra

Entropia = 5.809 bits

Por fim fizemos o teste de combinações de letras 3 a 3:

Tamanho total da palavra: 3669 (tamanho total da palavra /3)
nota: neste teste a divisão por 3 não dá resto 0 por isso o ultimo bit não entra para este teste.

Combinação AAA: 44 vezes, ou seja em 1.20 % da palavra
Combinação AAC: 30 vezes, ou seja em 0.82 % da palavra
Combinação AAG: 54 vezes, ou seja em 1.47 % da palavra
Combinação AAT: 32 vezes, ou seja em 0.87 % da palavra

Combinação ACA: 48 vezes, ou seja em 1.31 % da palavra
Combinação ACC: 54 vezes, ou seja em 1.47 % da palavra
Combinação ACG: 14 vezes, ou seja em 0.38 % da palavra
Combinação ACT: 43 vezes, ou seja em 1.17 % da palavra

Combinação AGA: 85 vezes, ou seja em 2.32 % da palavra
Combinação AGC: 63 vezes, ou seja em 1.72 % da palavra
Combinação AGG: 112 vezes, ou seja em 3.05 % da palavra
Combinação AGT: 50 vezes, ou seja em 1.36 % da palavra

Combinações ATA: 24 vezes, ou seja em: 0.65 % da palavra
Combinações ATC: 45 vezes, ou seja em: 1.23 % da palavra
Combinações ATG: 54 vezes, ou seja em: 1.47 % da palavra
Combinações ATT: 25 vezes, ou seja em: 0.68 % da palavra

Combinações CAA: 47 vezes, ou seja em: 1.28 % da palavra
Combinações CAC: 67 vezes, ou seja em: 1.83 % da palavra
Combinações CAG: 112 vezes, ou seja em: 3.05 % da palavra
Combinações CAT: 58 vezes, ou seja em: 1.58 % da palavra

Combinações CCA: 84 vezes, ou seja em: 2.29 % da palavra
Combinações CCC: 93 vezes, ou seja em: 2.53 % da palavra
Combinações CCG: 24 vezes, ou seja em: 0.65 % da palavra
Combinações CCT: 128 vezes, ou seja em: 3.49 % da palavra

Combinações CGA: 9 vezes, ou seja em: 0.25 % da palavra
Combinações CGC: 19 vezes, ou seja em: 0.52 % da palavra
Combinações CGG: 26 vezes, ou seja em: 0.71 % da palavra
Combinações CGT: 20 vezes, ou seja em: 0.55 % da palavra

Combinações CTA: 36 vezes, ou seja em: 0.98 % da palavra
Combinações CTC: 88 vezes, ou seja em: 2.40 % da palavra
Combinações CTG: 104 vezes, ou seja em: 2.83 % da palavra

Combinações CTT: 71 vezes, ou seja em: 1.94 % da palavra
Combinações GAA: 46 vezes, ou seja em: 1.25 % da palavra
Combinações GAC: 47 vezes, ou seja em: 1.28 % da palavra
Combinações GAG: 94 vezes, ou seja em: 2.56 % da palavra
Combinações GAT: 52 vezes, ou seja em: 1.42 % da palavra

Combinações GCA: 63 vezes, ou seja em: 1.72 % da palavra
Combinações GCC: 72 vezes, ou seja em: 1.96 % da palavra
Combinações GCG: 17 vezes, ou seja em: 0.46 % da palavra
Combinações GCT: 79 vezes, ou seja em: 2.15 % da palavra

Combinações GGA: 103 vezes, ou seja em: 2.81 % da palavra
Combinações GGC: 74 vezes, ou seja em: 2.02 % da palavra
Combinações GGG: 120 vezes, ou seja em: 3.27 % da palavra
Combinações GGT: 60 vezes, ou seja em: 1.64 % da palavra

Combinações GTA: 35 vezes, ou seja em: 0.95 % da palavra
Combinações GTC: 41 vezes, ou seja em: 1.12 % da palavra
Combinações GTG: 78 vezes, ou seja em: 2.13 % da palavra
Combinações GTT: 52 vezes, ou seja em: 1.42 % da palavra

Combinações TAA: 24 vezes, ou seja em: 0.65 % da palavra
Combinações TAC: 29 vezes, ou seja em: 0.79 % da palavra
Combinações TAG: 39 vezes, ou seja em: 1.06 % da palavra
Combinações TAT: 25 vezes, ou seja em: 0.68 % da palavra

Combinações TCA: 42 vezes, ou seja em: 1.14 % da palavra
Combinações TCC: 100 vezes, ou seja em: 2.73 % da palavra
Combinações TCG: 10 vezes, ou seja em: 0.27 % da palavra
Combinações TCT: 76 vezes, ou seja em: 2.07 % da palavra

Combinações TGA: 78 vezes, ou seja em: 2.13 % da palavra
Combinações TGC: 58 vezes, ou seja em: 1.58 % da palavra
Combinações TGG: 97 vezes, ou seja em: 2.64 % da palavra
Combinações TGT: 57 vezes, ou seja em: 1.55 % da palavra

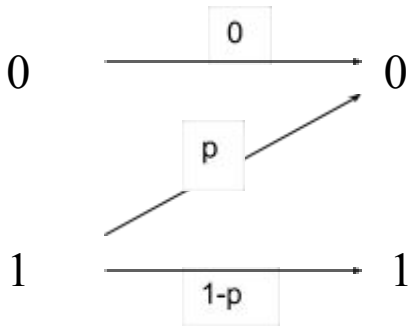
Combinações TTA: 29 vezes, ou seja em: 0.79 % da palavra
Combinações TTC: 60 vezes, ou seja em: 1.64 % da palavra
Combinações TTG: 69 vezes, ou seja em: 1.88 % da palavra
Combinações TTT: 80 vezes, ou seja em: 2.18 % da palavra

Entropia = 5.802 bits
nota: note-se que o valor da entropia tende a estabilizar por volta dos valores 5.80

Em termos de canal, o canal que encontramos é um canal do tipo Z (Z-channel ou canal binário assimétrico), ou seja, é um canal com entrada binária e saída binária onde o a transferência 1 → 0 ocorre com probabilidade não negativa p, enquanto o transferência 0 → 1 ocorre com probabilidade igual a 0.

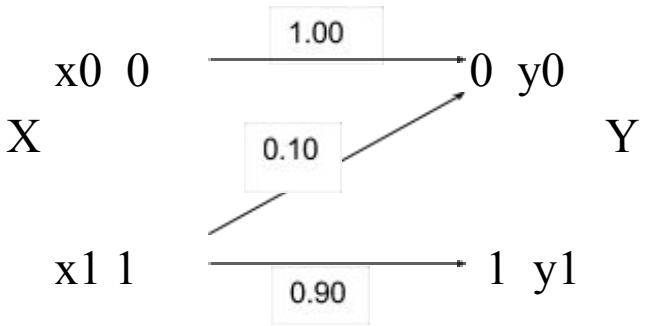
Ou seja, do tipo:

[1.00 p ;
0.00 1-p]



Sendo no caso do trabalho:

[1.00 0.10 ;
0.00 0.90]



Considerando X como antes e o Y como depois da passagem plo canal,
P(0) em X -> P(x0) = 1-α
P(1) em X -> P(x1) =α

P(0) em Y -> P(y0) = (1- α) + αp
P(1) em Y -> P(y1) = α(1-p)

I(X; Y) = H(Y) – H(Y| X)

1) Calculo de H(Y)

P(y0) = (1-α) + αp
P(y1) = α(1-p)

H(Y) = - (1- α + αp) log (1- α + αp) - α(1-p) log α(1-p)

---

2) Calculo de H(Y|X)

H(Y|X) = - αp log p – α(1-p) log (1-p)

---

3) I(X ; Y) = - (1- α + αp) log (1- α + αp) - α(1-p) log α + αp log p

---

$$\frac{\partial I(X;Y)}{\partial \alpha} = 0 \quad \text{-----} > \quad (1\text{-}p)\log(1-\alpha+\alpha p) - (1\text{-}p)\log\alpha + p\log p = 0$$

$$\alpha 0 = \frac{1}{(1-p+p^{\frac{p}{p-1}})}$$

$$C = -(1-\alpha 0+\alpha 0\, p)\log(1-\alpha 0+\alpha 0 p) - \alpha 0\,(1\text{-}p)\log\alpha 0 + \alpha 0\, p\log p =$$

$$C = -\log \alpha 0 - \frac{p}{(p-1)}\log p$$

$$C = \log(1-p+p^{(\frac{p}{p-1})}) - \frac{p}{p-1}\log p$$

Sabendo que p = p(x1 -> y0) , ou seja, probabilidade de erro de 1 passar para 0, podemos calcular a capacidade do canal:

Com p= 0.10

$$C = \log(1-0.10+0.10^{(\frac{0.10}{0.10-1})}) - \frac{0.10}{0.10-1}\log 0.10$$

Tem-se C = 0.762848

Tarefa 2:

A simulação foi feita do seguinte modo:

1. Aplicamos o algoritmo de Hamming (presente no ficheiro Hamming.java) a cada 4 bits da palavra de entrada.
2. O que gera um output transformado de mais 3 bits (3 check bits).
3. De seguida introduzimos um erro cada vez que encontramos um bit a '1'.
4. A maneira de introduzir o erro é fazer um random de 1 a 10, quando esse random atinge o numero 1 (neste caso escolhemos 1 porque calhou, podia ser um outro qualquer, teria era de ser sempre o mesmo para o erro ser de 10%).
5. Depois de introduzidos os erros, a palavra é enviada para descodificar, desta vez 7 a 7 bits e cada 7 bits transformados de volta a 4 bits (agora sem os 3 check bits inseridos anteriormente).
   nota: se houver mais do que um erro inserido a cada 7 bits, a correção de erro do hamming vai falhar e vai corrigir no sitio errado, podendo corromper a maior parte da palavra.
6. Finalmente, depois da correção feita (com erros ou não) é feito o envio da palavra para o ficheiro de texto inserido ao correr o programa.

nota: Os 3 check bits utilizados para fazer a codificação de Hamming foram encontrados fazendo XOR's com os 4 data bits originais(check1 = data1 XOR data2 XOR data4, check2 = data1 XOR data3 XOR data4 e check3 = data2 XOR data3 XOR data4).

Tarefa 3:

Método para correr em java:

1. Compilar com o javac (caso não seja possível serão incluídos na pasta os ficheiros já compilados)
2. Correr como parâmetro, visto que de outra maneira não funciona.
   nota: os programas tem de ser corridos com java antes do nome do programa e dos parametros (java code input.txt > output.txt) visto que o compilador do java não cria executaveis como o compilador do C.

Programa "code":

1. Este programa recebe como input a string com a palavra que queremos comprimir e codificar.
2. Depois de a receber, comprime-a utilizando o algoritmo LZW (presente no ficheiro LZW.java) dado nas aulas.
3. Este algoritmo comprime a palavra para um array com cerca de 2406 indexes e faz o output do dito array.
4. Este array é depois codificado com codigo binario (até á posição 32 da tabela é codificado com 5 bits, depois passa para 6 bits até a posição 64 e etc até chegar á posição 2048 que codifica com 12 bits, isto porque até á posição 32 do array o maior numero utilizado só vai precisar de 5 bits para ser codificado, até ao 64 só precisa de 6 bits e tc).
5. Depois de codificado o codigo da lista é feita uma concatenação de todas as posições da lista e a string resultante é enviada para o ficheiro de output

Programa "decode":

1. Este programa faz o contrario do programa "code". Recebe uma string codificada e "desconcatena-a" para uma lista de indexes.
2. Com esta lista, descodifica-a com um parseInt do java simples, passando de binário para inteiro.
3. Depois de ter a lista com os inteiros procede á descompressão (do método decompress do LZW).
4. Finalmente retorna a string para o stdout para escrever para o ficheiro.
   nota: faz ainda um write para o ficheiro da quantidade de erros encontrados durante a descompressão.

Conclusão:

Antes da compressão o ficheiro tinha, cerca de, 11mil caracteres, depois da compressão e codificação fica com um total de, cerca de, 24mil caracteres.
Isto deve-se ao facto de a lista de indexes criada ser muito grande, e a conversão dos inteiros para binário ficar com um tamanho de bit máxima de 12 , dando um total de, aproximadamente, 24mil bits (O que se codificando cada carácter com 2 bits no ficheiro original daria um total de 22mil, neste caso seria melhor do que a codificação escolhida)

O ficheiro final (seq-output.txt) não contem uma palavra igual á palavra passada inicialmente ao programa “code”. Isto é devido aos erros durante a correção de erros, visto que o Hamming só consegue corrigir 1 erro de cada vez, se uma palavra contiver mais do que um erro, este corrigira na posição errada, e a palavra final ficará corrompida.