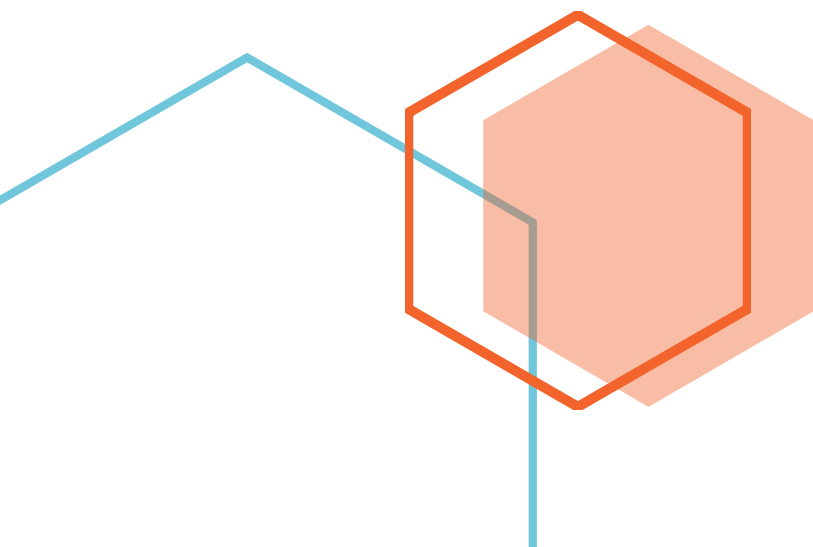


Teoria da Informação

Trabalho prático nº1

Ana Carolina Morais - 2021222056
Fernanda Fernandes – 2021216620
Inês Quintal - 2021232993



Índice

Introdução	2
Fundamentos da teoria da informação – Conceitos aplicados	2
Exercício 1 – Histograma de Ocorrência dos seus símbolos.....	4
Exercício 2 – Cálculo da entropia.....	4
Exercício 3	4
Pergunta: Será possível comprimir cada uma das fontes de forma não destrutiva?	6
Exercício 4 – Cálculo da entropia usando a Codificação de Huffman.....	7
Pergunta: Será possível reduzir-se a variância?	8
Exercício 5 – Análise de dados com agrupamento de símbolos.....	8
Exercício 6 – Cálculo da informação mútua, utilizando uma janela deslizante	9
Conclusão	14

Introdução

O trabalho prático tem como principal objetivo adquirir sensibilidade para as questões fundamentais de teoria da informação, em particular informação, redundância, entropia e informação mútua.

Como tal, os conceitos fundamentais de teoria da informação vão ser aplicados a diversas fontes, tais como: imagens, som e texto.

Para o desenvolvimento do código, utilizámos o ide *Spyder* (versão 4.2.5). Através do mesmo garantimos que todo o código cumpre os requisitos associados ao *Python*. Ao utilizarmos esta linguagem, temos como vantagem a utilização de bibliotecas tais como: *Numpy*, *Matplotlib*, *Scipy.io.wavfile*.

Ao longo deste trabalho são usados alguns conceitos teóricos, (que serão mais tarde explicitados) tais como: Entropia, Informação Mútua, Código de Huffman e Árvores de Huffman.

Fundamentos da teoria da informação – Conceitos aplicados

Entropia – Número médio de bits para codificar uma fonte de informação.

$$H(A) = \sum_{i=1}^n P(a_i) i(a_i) = - \sum_{i=1}^n P(a_i) \log_2 P(a_i)$$

Legenda:

H = Entropia;

$P(a(i)) = P(A = a(i))$ = Probabilidade de um dado elemento do alfabeto;

n = Número de elementos de um dado alfabeto;

Entropia Máxima – Número de bits por símbolo necessários para codificar uma fonte, sem que esta esteja comprimida e admitindo que as probabilidades são equiprováveis.

$$H_{max}(X) = \sum_i \log_2 |A_x|$$

Com o objetivo de diminuir o espaço ocupado por um conjunto de dados num determinado dispositivo é feita, anteriormente, uma compressão desses mesmos dados.

A codificação em Códigos de Huffman é uma forma de compressão de dados, baseando-se nas probabilidades de ocorrência de símbolos e, posteriormente realocados no conjunto de símbolos. Por fim, a codificação termina, quando todos os símbolos estiverem unidos a símbolos auxiliares, de forma a que a probabilidade dele seja 1, formando assim uma árvore binária.

Neste método é atribuído menos bits aos símbolos mais frequentes e mais bits aos símbolos de menor frequência.

Informação Mútua – Quantidade de informação que uma variável contém sobre outra.

$$\begin{aligned} I(X, Y) &= \sum_{x \in A_x} \sum_{y \in A_y} P(x, y) \log_2 \frac{P(y | x)}{P(y)} \\ &= \sum_{y \in A_y} P(y) \log_2 \frac{1}{P(y)} - \sum_{x \in A_x} \sum_{y \in A_y} P(x, y) \log_2 \frac{1}{P(y | x)} \\ &= H(Y) - H(Y | X) \end{aligned}$$

Exercício 1 – Histograma de Ocorrência dos seus símbolos.

Neste exercício, criámos 3 funções: *texto ()*, *audio_imagem ()* e *desenhar_histograma ()*, cujo objetivo das primeiras duas é calcular o número de ocorrências de uma letra num ficheiro de texto/imagem/áudio. O número de ocorrências de um ficheiro de imagem é calculado da mesma forma que um ficheiro de áudio.

Exercício 2 – Cálculo da entropia

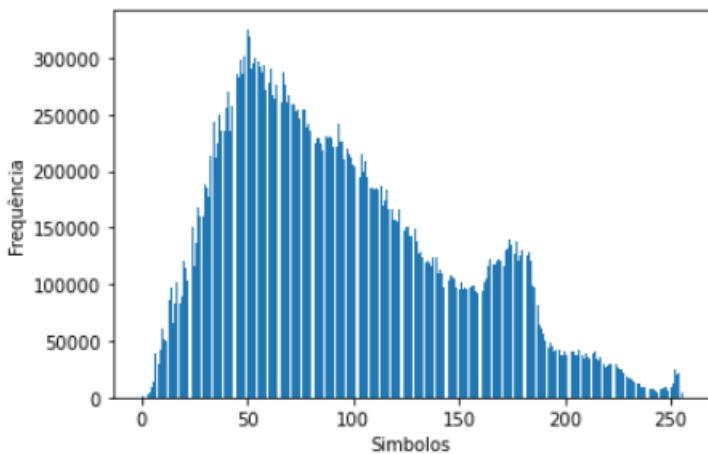
Utilizando o que aprendemos nas aulas teóricas e, que apresentámos a fórmula acima, criámos duas funções. Uma auxiliar, que devolve a probabilidade de ocorrência de um símbolo. E outra que calcula a entropia.

Exercício 3 – Análise de dados

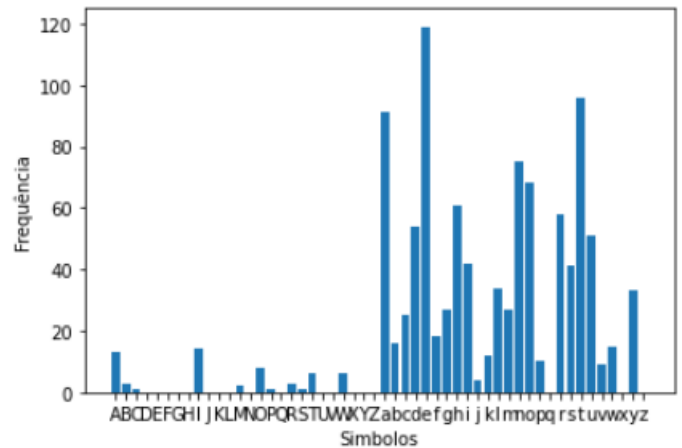
Neste exercício, recorrendo aos exercícios anteriores, determinámos e visualizámos a distribuição estatística e a entropia de várias fontes. Vamos agora, visualizar na tabela abaixo a entropia de cada ficheiro.

Ficheiro	Entropia (Bits/Símbolo)
landscape.bmp	7.606945
lyrics.txt	4.410705
MRI.bmp	6.860541
MRIbin.bmp	0.661080
soundMono.wav	4.065729

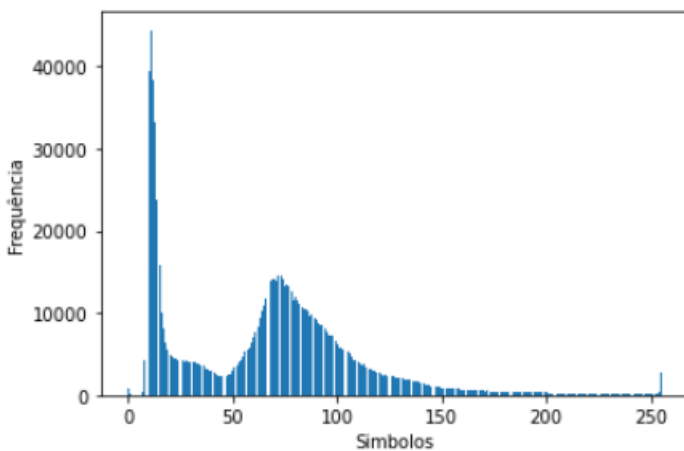
Tabela 1

1. "landscape.bmp"

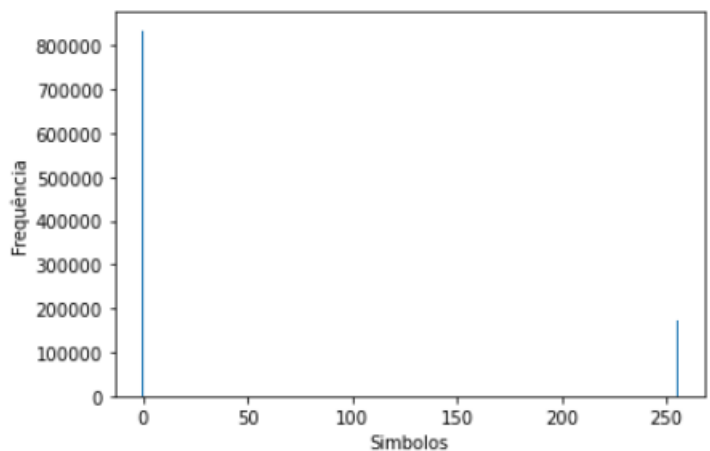
Esta imagem revela um maior valor de entropia, uma vez que é uma imagem em tons de cinzento (pixéis variam entre 0 e 255). Os valores estão distribuídos de forma uniforme.

2. "lyrics.txt"

Nesta fonte (texto) analisámos que existe alguma dispersão, dado que as letras minúsculas ocorrem com mais frequência que as maiúsculas.

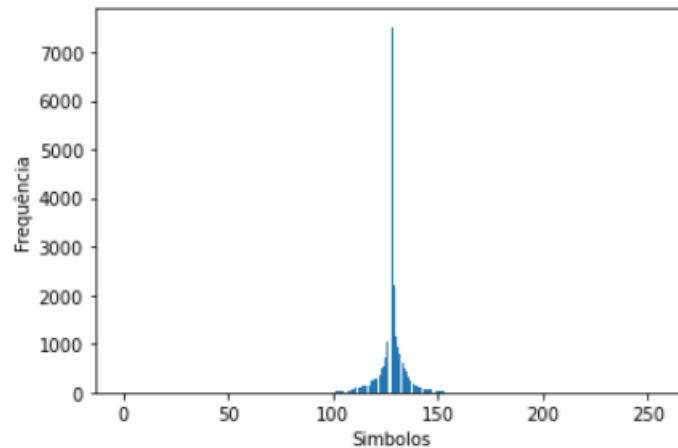
3. "MRI.bmp"

Nesta fonte, existe uma maior diferença de valores dependendo das zonas do gráfico. É de reparar que existe os símbolos iniciais são mais recorrentes que os finais.

4. "MRIbin.bmp"

Esta imagem apresenta o menor valor de entropia, uma vez que se trata de uma imagem binária, cujo alfabeto contém apenas 2 símbolos (0-Preto/255-Branco).

5. "soundMono.wav"



Nesta fonte (som), observa-se uma simetria no centro, havendo maior concentração nesse ponto e, menor concentração à sua volta.

Pergunta: Será possível comprimir cada uma das fontes de forma não destrutiva?
Se sim, qual a compressão máxima que se consegue alcançar?

Primeiro, é necessário explicar o que é a compressão de dados, portanto a compressão de dados é o ato de reduzir o espaço ocupado por dados num determinado dispositivo. Para comprimir dados é de extrema importância reduzir a redundância, uma vez que muitos dados contêm informações redundantes que podem, ou precisam de ser eliminadas de alguma forma. Assim, garantimos que os dados comprimidos serão idênticos aos dados originais, de forma a que se realizamos o processo inverso iremos obter os dados originais.

Respondendo à questão, sim, é possível comprimir cada uma das fontes de forma não destrutiva, uma vez que, os valores de entropia obtidos são, sempre, inferiores ao número de bits necessários para codificar cada símbolo. Para o cálculo da taxa de compressão máxima, é necessário utilizar a seguinte fórmula:

$$TC_{max} = \frac{H_{max}(X) - H(X)}{H_{max}(X)} * 100\%$$

Onde

$$H_{max} = \log_2(k)$$

Corresponde a entropia máxima para uma fonte com um alfabeto de (k) símbolos.

Após a aplicação da fórmula anterior, colocámos a informação toda na seguinte tabela:

Ficheiro	Entropia (bits/Símbolo)	Entropia máxima(bits/símbolo)	Taxa de Compressão (%)
landscape.bmp	7.606945	8 ¹	4,913575
lyrics.txt	4.410705	$\log_2 (52)^2$	22,6252
MRI.bmp	6.860541	8	14,24324
MRIbin.bmp	0.661080	8	91,7365
soundMono.wav	4.065729	8	49,17839

Tabela 2

Exercício 4 – Cálculo da entropia usando a Codificação de Huffman

Ficheiro	Entropia (bits/Símbolo)	Entropia (Huffman) (bits/símbolo)	Variância
landscape.bmp	7.606945	7.629310	0.751665
lyrics.txt	4.410705	4.443492	1.082083
MRI.bmp	6.860541	6.890985	2.193083
MRIbin.bmp	0.661080	1,000000	0,000000
soundMono.wav	4.065729	4.11078	4.355634

Tabela 3

Após observar os resultados obtidos da tabela 3, concluímos que a entropia calculada anteriormente apresenta valores muito próximos do cálculo da entropia usando a codificação de Huffman, sendo estes últimos superiores.

¹ O número é 8, uma vez que existem 256 símbolos (0-255) no alfabeto, que é $\log_2 (256) = 8$.

² O número é 52, uma vez existem 52 símbolos no alfabeto (26- maiúsculas + 26-minúsculas).

Contudo, observámos que respeitam a seguinte condição:

$$H(S) \leq \bar{l} < H(S) + 1$$

\bar{l} -> número médio de bits/Símbolo da codificação de Huffman

Relativamente, à variância, que é uma medida de dispersão, em relação à média, ou seja, indica-nos a “distância” dos valores dos códigos de Huffman para o número médio de bits/Símbolo (entropia). Cujas fórmulas, foi dada na cadeira de “Estatística”.

$$V(X) = E(X^2) - (E(X))^2 \quad (\text{fórmula de Koenig})$$

Pergunta: Será possível reduzir-se a variância?

Se Sim, como pode ser feito e em que circunstância será útil?

Sim, é possível reduzir a variância, se criarmos árvores de Huffman com menor profundidade, ou seja, se reagruparmos os símbolos por ordem mais elevada possível, de forma a que fiquem o mais próximo possível da raiz.

Ao pôr-mos em prática esta solução, podemos contornar o congestionamento de rede numa transmissão de dados, dado que reduz a capacidade de transmissão possível da rede necessária para transmitir o maior símbolo.

Exercício 5 – Análise de dados com agrupamento de símbolos

No exercício 5, implementamos a função do cálculo da entropia, utilizada anteriormente, mas agora com o agrupamento de símbolos. Para essa realização foi necessário criar duas novas funções para tratamento de texto/imagem/som (como já foi averiguado, o alfabeto de uma fonte de texto é diferente de uma fonte de som/imagem, daí duas funções diferentes).

Para o tratamento numa fonte de texto, decidimos criar um novo alfabeto, em que se usam números em vez de letras, e ainda cada número corresponde a um par de letras.

Por fim sabendo a posição onde incrementar o *array* de ocorrências, usámos o mesmo para o cálculo da entropia (divido por 2, pois agrupámos em pares).

Ficheiro	Entropia (bits/Símbolo)	Entropia conjunta (bits/Símbolo)
landscape.bmp	7.606945	6.204059
lyrics.txt	4.410717	3.652207
MRI.bmp	6.860487	5.199170
MRIbin.bmp	0.661079	0.402871
soundMono.wav	4.065812	3.310445

Tabela 4

Após a análise dos dados da tabela, concluímos que a entropia diminuiu, quando agrupámos os símbolos dois a dois. Uma das vantagens da realização do agrupamento é o facto, de tornar o programa mais eficiente no que toca à dispersão da distribuição. Já uma das desvantagens, é o facto de o comprimento do alfabeto aumentar consideravelmente, o que é um problema para a memória.

Exercício 6 – Cálculo da informação mútua, utilizando uma janela deslizante

Exercício a) Cálculo do vetor de valores de informação mútua

Neste exercício, criámos uma função que percorre o sinal onde pesquisar (*target*), de forma a conseguir isolar uma janela, do mesmo tamanho que o sinal a pesquisar (*query*), sendo possível calcular a informação mútua, dado que a janela irá “deslizar” sobre o *target*, até este chegar ao fim. Para isso, utilizámos a fórmula de informação mútua enunciada nos conceitos utilizados.

Exercício b) Cálculo da informação mútua (*Target* e *Query* diferentes da alínea anterior)

Neste exercício, aplicámos a mesma teoria da alínea a), mas agora utilizando o ficheiro “saxriff.wav” como *query* e os ficheiros “target01 - repeat.wav” e “target02 - repeatNoise.wav” como *targets*. Outra alteração, é que na alínea anterior utilizámos o passo = 1 e, neste caso, o passo = ($\frac{1}{4}$ * comprimento do *query*). Vamos agora analisar a evolução da informação mútua nas figuras abaixo.

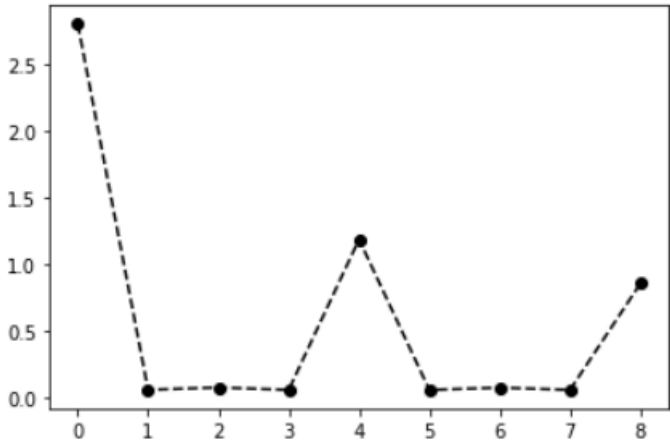
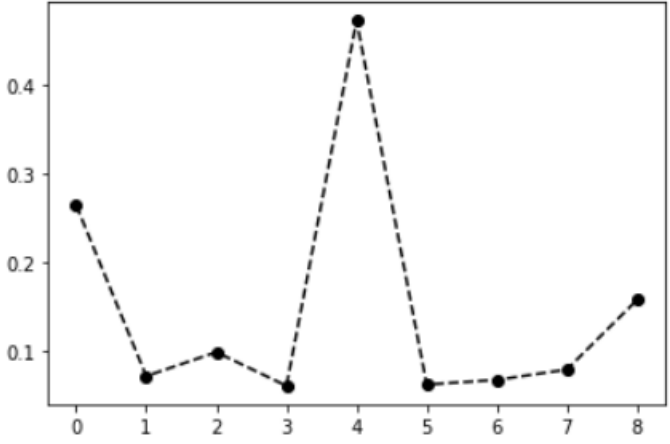
Target01 - repeat.wav	Target02 - repeatNoise.wav
[2.8027, 0.0618, 0.0793, 0.0612, 1.1882, 0.0608, 0.0781, 0.0611, 0.8573]	[0.2647, 0.0716, 0.0986, 0.0606, 0.4727, 0.0621, 0.0673, 0.079, 0.1576]
	

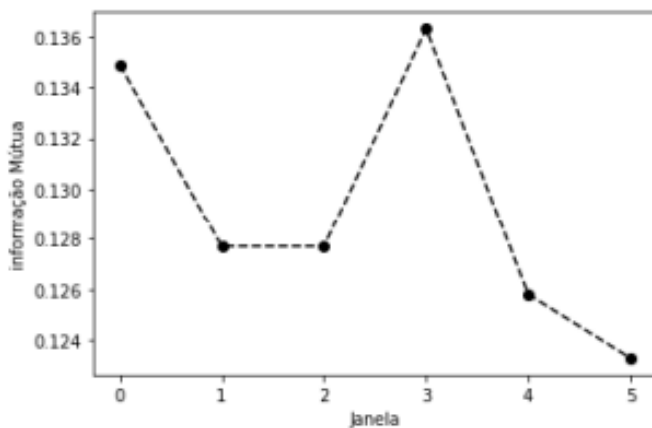
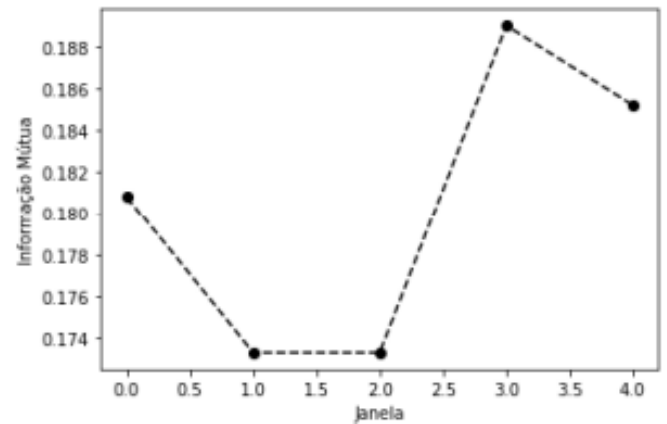
Tabela 5

Através da análise da tabela, podemos concluir que no primeiro gráfico, o primeiro ponto se destaca mais, uma vez que corresponde à primeira janela. O que se verifica, pois o "Target01 - repeat.wav" corresponde a várias repetições do "saxriff.wav", isto é, os sons são quase iguais, numa primeira fase inicial. À medida que se avança no target, os valores da informação mútua vão se diminuindo, dado que os sons deixam de coincidir na totalidade. Olhando para o segundo gráfico, uma vez que há presença de ruído será de esperar uma diminuição dos valores de informação mútua.

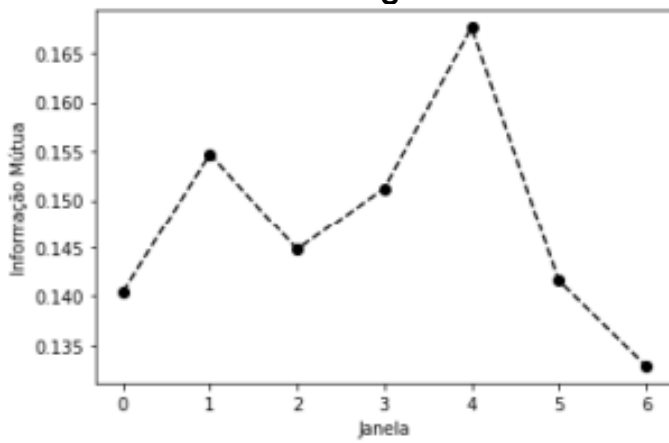
Exercício c) Extensão da alínea anterior aplicado a vários sons

Ficheiro	Informação Mútua	Informação Mútua máxima
Song01	[0.1349, 0.1277, 0.1277, 0.1363, 0.1258, 0.1233]	[0.1363]
Song02	[0.1808, 0.1733, 0.1733, 0.189, 0.1852]	[0.189]
Song03	[0.1676, 0.1547, 0.1512, 0.1448, 0.1416, 0.1404, 0.1329]	[0.1676]
Song04	[0.1898, 0.1906, 0.1897, 0.1903, 0.1898, 0.1896]	[0.1906]
Song05	[0.5322]	[0.5322]
Song06	[3.531]	[3.531]
Song07	[3.531]	[3.531]

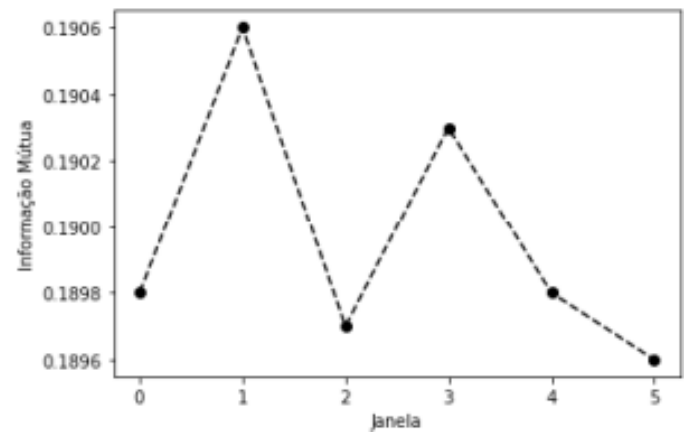
Tabela 6

1.Song.01**2.Song.02**

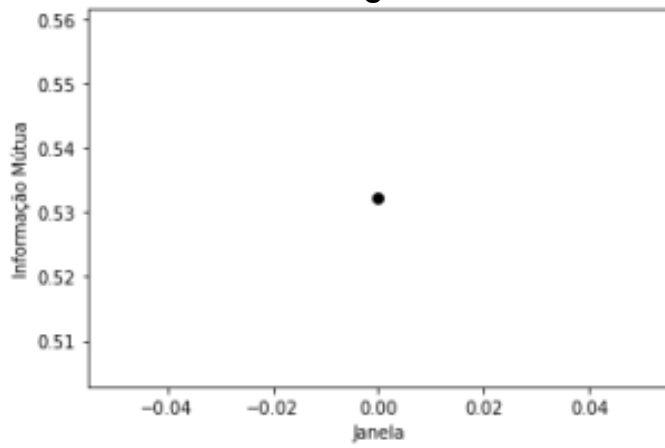
3.Song.03



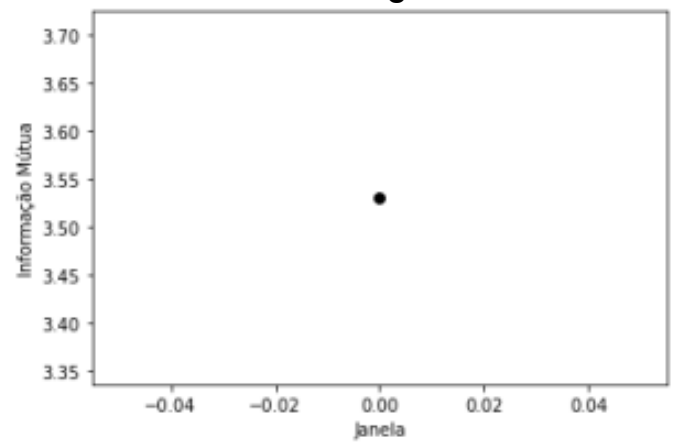
4.Song.04



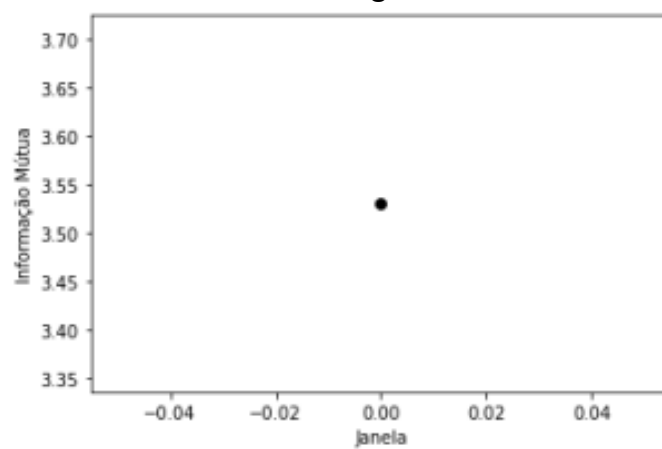
5.Song.05



6.Song.06



7.Song.07



Após a análise da tabela 6, é fácil retirar o valor da informação mútua máxima, através da lista de informações mútuas de casa som. Também, uma das vantagens desta tabela é perceber qual/ quais dos sons são parecidos ao *query*.

Nota: Definimos o passo = $\frac{1}{4}$ como está no enunciado, uma vez que este vai percorrer mais vezes o *target*, ou seja, maior será a quantidade de janelas a ser comparada com o *target*, ao comparar este passo com o passo da simulação conseguimos destacar essa mesma diferença entre janelas.

Quanto mais janelas percorrem o som, mais valores de informação mútua vão aparecendo. Isto deve-se ao facto de os sons terem dimensões diferentes ao *query*, como é o caso dos sons 01 a 04. No caso dos sons 05 a 07, só apresentam um valor de informação mútua, logo uma janela, por terem dimensões semelhantes ao *query*.

Ao observar os gráficos dos sons 06 e 07, conseguimos afirmar que a informação mútua é a mesma.

Por último, vamos agora ver a tabela com os valores de informação mútua por ordem decrescente, como pedia no enunciado.

Ficheiro	Informação Mútua
Song06	[3.531]
Song07	[3.531]
Song05	[0.5322]
Song04	[0.1906]
Song02	[0.189]
Song03	[0,1676]
Song01	[0,1363]



Conclusão

Em suma, ao realizarmos este trabalho prático, conseguimos por em prática os demais conceitos ensinados na teórica e, presenciar uma mini experiência de como funciona o “Shazam”.