Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Algorithmic Information Theory (2021/22)

Lab work n.º 3

Grupo n.º 10

Fernando Lopes n.º 106358

Maria João Sousa n.º 109488

João Carvalho n.º 89059

Índice

Introdução

Neste trabalho é abordado como problema principal a semelhança entre ficheiros de áudio de forma a classificar os ficheiros como iguais ou distintos. Para a resolução deste problema foram usados quatro compressores pré existentes (*lzma, gzip, bz2 e zlib*) e uma junção dos 4.

A ideia principal é usar compressores pré existentes para comprimir os ficheiros descritivos das frequências, gerados pelo *getMaxFreqs*, e calcular o NCD a partir do espaço ocupado pelos ficheiros comprimidos. O par de ficheiros que tenha o valor de NCD mais baixo, corresponde ao par de ficheiros mais similar.

Todos os passos de implementação, ideias de desenvolvimento e análise dos resultados serão descritos ao longo do relatório.

Análise da Implementação

Para a resolução do problema proposto foi, numa primeira fase, criada uma base de dados com excertos de músicas(25 no total) dos mais variados géneros guardada no diretório "*Samples*", quase todas com 30 segundos de duração. Seguidamente foram criados ficheiros de áudio com o objetivo de testar o programa com excertos dos *samples* referidos. Optou-se por formar quatro diferentes tipos de testes quanto à sua duração: 1, 3, 6 e 10 segundos. Testes estes que se encontram nos diretórios "*Test_1s*", "*Test_3s*", "*Test_6s*", "*Test_10s*" respectivamente. Além dos *samples* e dos ficheiros de teste, foi também criado um diretório com ficheiros áudio de ruídos, de entre os ruídos escolhidos encontram-se efeitos sonoros no geral, ruído de frequência, aplausos, gritos etc. Estes últimos ficheiros de áudio tem como objetivo realizar testes mais complexos e desafiadores ao programa, para isso concatenou-se os ficheiros de ruído com os de teste(sobreposição de áudios). Note-se que os ficheiros de ruído não devem ser mais extensos que os de teste, pois, caso contrário, uma vez concatenados, o teste final ficaria também mais extenso e sairia do padrão de duração(1,3,6,10 segundos).

Para o processamento dos áudios(retirar excertos, concatenação) foi usado o software *Sound eXchange(SoX)*[3].

O programa principal está contido no ficheiro *main.py* onde se consegue perceber o funcionamento do sistema, nomeadamente o problema de identificação de um determinado ficheiro de teste. Além disso, também está presente o ficheiro *evaluation.py* onde se levou a

cabo uma avaliação do sistema através de algumas demonstrações explicadas e analisadas no bloco de resultados mais adiante.

Funcionamento do programa

À semelhança de outros softwares que têm como objetivo permitir ao utilizador descobrir que música é que o utilizador está a ouvir, como o *Shazam* ou o *SoundHound*, o utilizador apenas necessita de ter um ficheiro *.wav* da música que pretende categorizar(ficheiro teste).

O sistema utiliza quatro modos de compressão, que podem ser especificados pelo utilizador, são eles:

- LZMA
- Gzip
- Zlib
- Bz2
- All - utiliza os todos os compressores para categorizar a música

Como veremos mais abaixo, o compressor que obteve uma melhor precisão de resultados foi o *gzip* então foi colocado como o compressor *default*.

Para o programa poder categorizar dois ficheiros como semelhantes ou não, é calculado o NCD para cada par ficheiro teste-ficheiro *sample*. O NCD é calculado da seguinte forma:

$$\mathrm{NCD}(x,y) = \frac{C(x,y) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}},$$

Em que C(x,y) representa o número de bits após o compressor escolhido comprimir a concatenação(das strings) de um ficheiro que contêm as frequências do ficheiro teste(x) e uma das *samples*(y), min{C(x), C(y)} representa o número de bits do ficheiro comprimido que ocupar menos espaço e max{C(x), C(y)} representa o número de bits do ficheiro comprimido que ocupar mais espaço.

Quanto menor é o valor do NCD para o par de ficheiros, menor é a diferença entre os ficheiros, logo mais probabilidade desse determinado ficheiro teste representar esse *sample*.

O modo de compressão All é diferente dos restantes uma vez que é calculado o NCD para todos os compressores e depois as posições em que esse par de ficheiros ficou são somadas e é devolvido o par que tiver a soma de posições menor. Este modo é mais bastante mais lento que os restantes, mas têm como objetivo melhorar os resultados de precisão do programa, uma vez que não depende apenas de um compressor.

Resultados e Avaliação

Neste bloco serão apresentadas algumas demonstrações de testes realizados ao programa com o objetivo de avaliá-lo. Consequentemente, será feita a análise dos resultados obtidos.

Os testes foram realizados no ficheiro *evaluation.py* e tem por base três variaveis:

- ***Test File Length***
  - Exprime a duração de áudio do ficheiro de teste. Contém exemplos com 1, 3, 6 e 10 segundos.

- ***Compressor***
  - Define o tipo de compressor. Contém os tipos anteriormente referidos.

- ***Noise***
  - Define se se realiza testes com ou sem presença de ruído.

A avaliação consiste em testar uma série de *test files* filtrados pelas três variáveis explanadas acima. Para obter uma avaliação de eficácia do programa foi usada a métrica ***Accuracy(%)***. Para o cálculo da mesma, simplesmente divide-se o número de acertos do programa pelo número total de testes realizados.

Accuracy para Diferentes Tipos de Compressores

Neste ponto serão expostos testes de avaliação dos tipos de compressor para cada duração dos ficheiros de teste. Nas figuras de 1 a 8, estão representados os gráficos com a variação de *accuracy* assim como os seus resultados exatos no terminal.

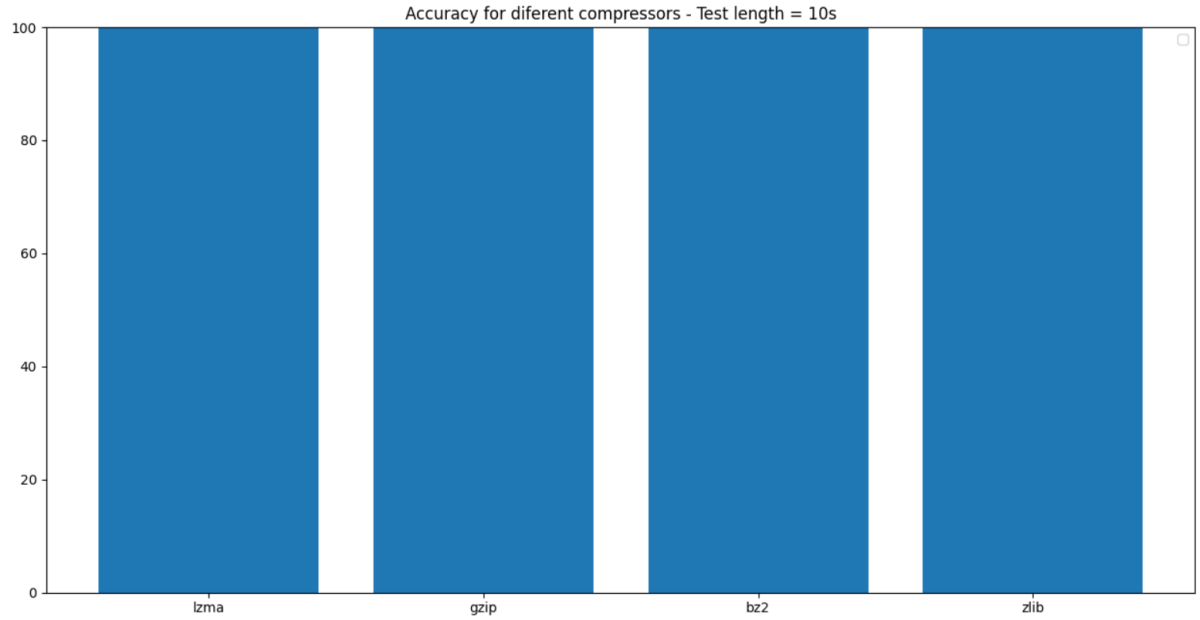

Fig. 1: Gráfico com *Accuracy* para diferentes compressores com testes de *length* 10

```
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 10s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: lzma | Length of the test file 10s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: gzip | Length of the test file 10s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: bz2 | Length of the test file 10s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 10s
Time: 6.214974069595337 min
```

Fig. 2: *Print*(terminal) com *Accuracy* para diferentes compressores com testes de *length* 10

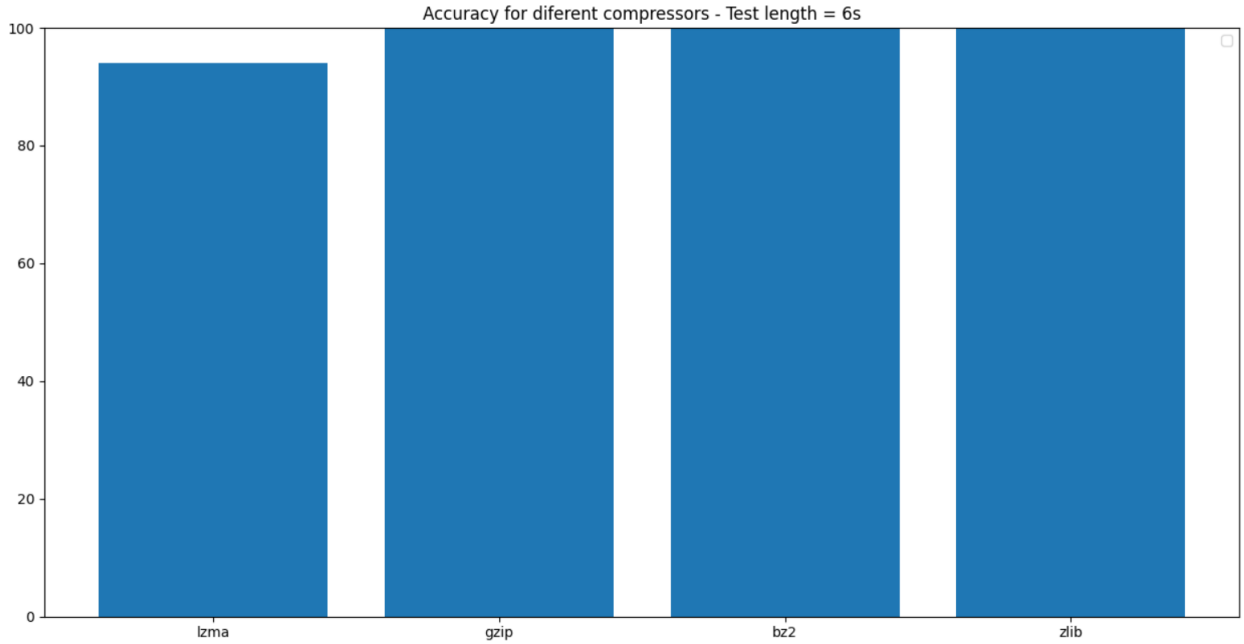

Fig. 3: Gráfico com *Accuracy* para diferentes compressores com testes de *length* 6

```
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 6s
Accuracy: 94.0% | 50 Tests without noise | Type of Compression: lzma | Length of the test file 6s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: gzip | Length of the test file 6s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: bz2 | Length of the test file 6s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 6s
Time: 5.860856334368388 min
```

Fig. 4: *Print*(terminal) com *Accuracy* para diferentes compressores com testes de *length* 6

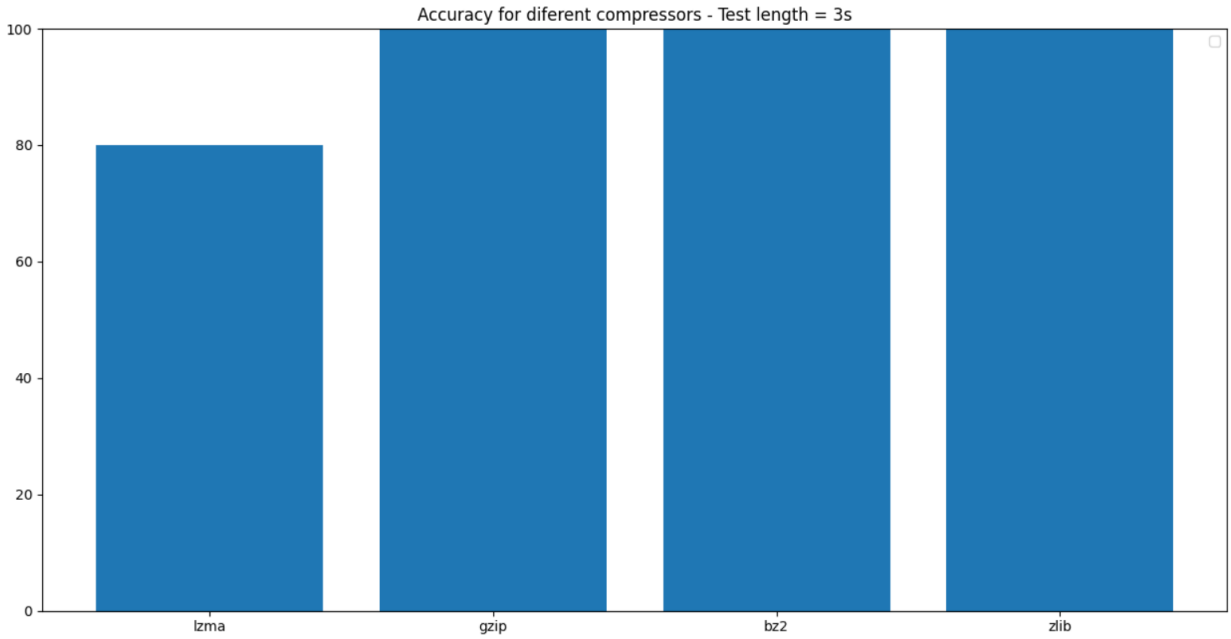

Fig. 5: Gráfico com *Accuracy* para diferentes compressores com testes de *length* 3

```
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 3s
Accuracy: 80.0% | 50 Tests without noise | Type of Compression: lzma | Length of the test file 3s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: gzip | Length of the test file 3s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: bz2 | Length of the test file 3s
Accuracy: 100.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 3s
Time: 5.746504525343577 min
```

Fig. 6: *Print*(terminal) com *Accuracy* para diferentes compressores com testes de *length* 3

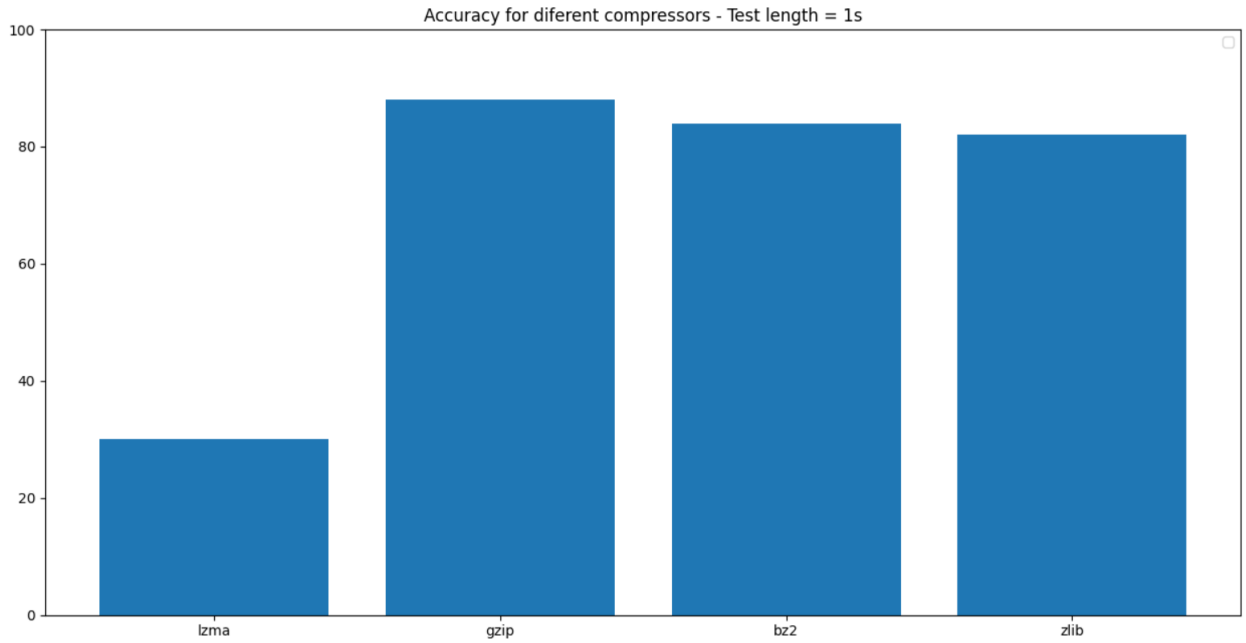

Fig. 7: Gráfico com *Accuracy* para diferentes compressores com testes de *length* 1

```
Accuracy: 82.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 1s
Accuracy: 30.0% | 50 Tests without noise | Type of Compression: lzma | Length of the test file 1s
Accuracy: 88.0% | 50 Tests without noise | Type of Compression: gzip | Length of the test file 1s
Accuracy: 84.0% | 50 Tests without noise | Type of Compression: bz2 | Length of the test file 1s
Accuracy: 82.0% | 50 Tests without noise | Type of Compression: zlib | Length of the test file 1s
Time: 7.782952892780304 min
```

Fig. 8: *Print*(terminal) com *Accuracy* para diferentes compressores com testes de *length* 1

Observando os resultados pode-se concluir que existe uma diferença considerável de eficácia do compressor *lzma* para os restantes, uma vez que apresenta uma *accuracy* bastante mais baixa quando comparada à média geral. Por outro lado, verificou-se uma ligeira vantagem do compressor *gzip* para os demais, diferença que é perceptível no gráfico de testes com 1 segundo(Fig. 7). Pode-se concluir, também, que quanto mais extensa for a amostra de teste, maior é a precisão.

Comparação com/sem *Noise* para as Diferentes Durações de Teste

Outro tipo de teste realizado foi a comparação de valores de *accuracy* de testes com e sem presença de *noise*. Para isso usou-se as diferentes durações de áudio dos testes como variável independente. Como já referido e comprovado acima, o compressor mais eficiente foi o *Gzip* e o menos eficiente o *Lzma*, então, por causa deste dado, foram usados estes dois compressores nesta demonstração.

Abaixo, nas figuras de 9 a 12, encontram-se os resultados obtidos.

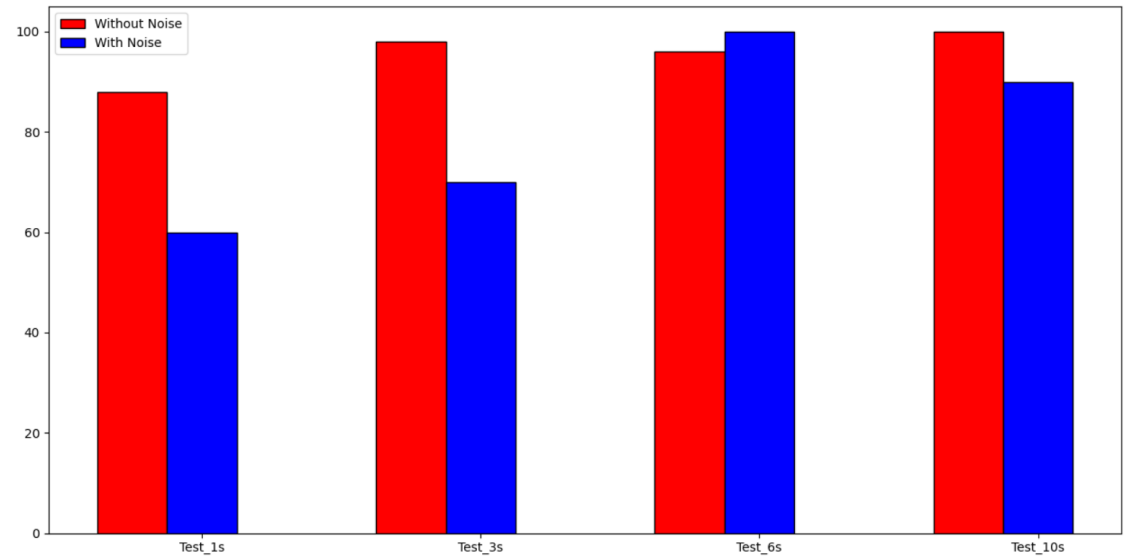

Fig. 9: *Gráfico* com *Accuracy* para diferentes durações de teste c/s *noise* - *gzip*

```
Without noise - Accuracy: 88.0% | 50 Tests | Type of Compression: gzip | Length of the test file 1s
With noise    - Accuracy: 60.0% | 10 Tests | Type of Compression: gzip | Length of the test file 1s
Without noise - Accuracy: 98.0% | 50 Tests | Type of Compression: gzip | Length of the test file 3s
With noise    - Accuracy: 70.0% | 10 Tests | Type of Compression: gzip | Length of the test file 3s
Without noise - Accuracy: 96.0% | 50 Tests | Type of Compression: gzip | Length of the test file 6s
With noise    - Accuracy: 100.0% | 10 Tests | Type of Compression: gzip | Length of the test file 6s
Without noise - Accuracy: 100.0% | 50 Tests | Type of Compression: gzip | Length of the test file 10s
With noise    - Accuracy: 90.0% | 10 Tests | Type of Compression: gzip | Length of the test file 10s
Time: 15.047262064615886 min
```

Fig. 10: *Print*(terminal) com *Accuracy* para diferentes durações de teste c/s *noise* - *gzip*

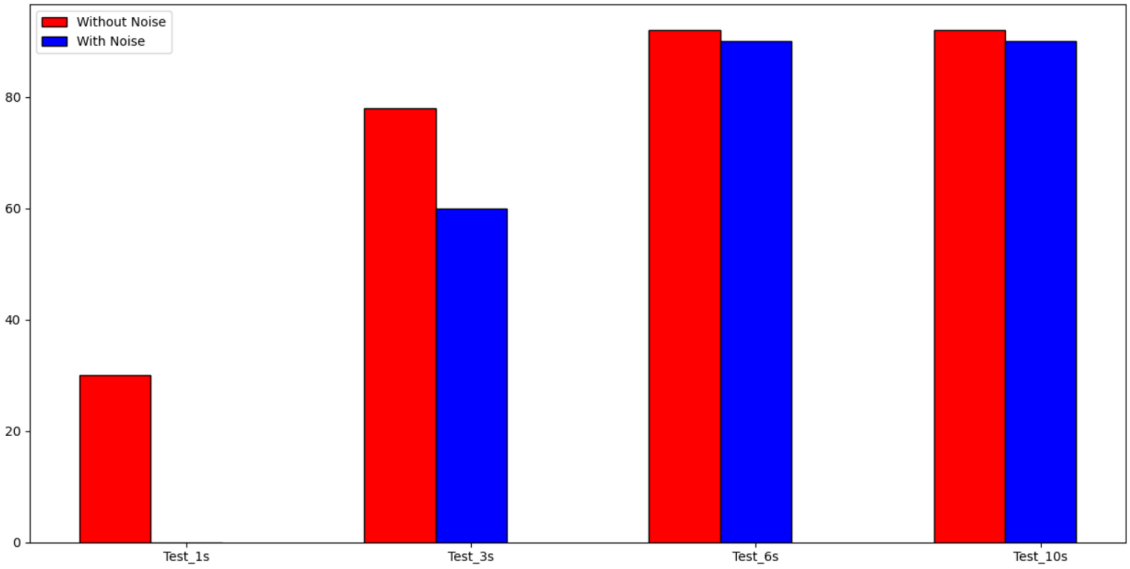

Fig. 11: *Gráfico* com *Accuracy* para diferentes durações de teste c/s *noise* - *lzma*

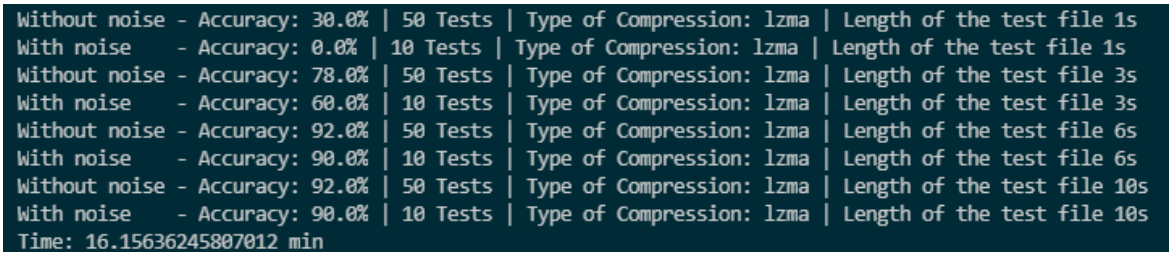

Fig. 12: *Print*(terminal) com *Accuracy* para diferentes durações de teste c/s *noise* - *lzma*

No seguimento da referência anterior, confirmou-se que o compressor *Gzip* tende a ser mais eficiente que o *Lzma*. Os resultados tendem a ser melhores, quanto mais longa for a amostra de áudio. A presença de noise tende a fazer diminuir a *accuracy*, especialmente para amostras mais curtas.

Obviamente estes comportamentos não se verificam na totalidade dos testes realizados como por exemplo, na figura 9, percebe-se que a *accuracy* foi mais elevada no teste de 3 segundos do que no de 6 segundos, o que contradiz o comportamento esperado, contudo à algumas variáveis que podem fazer isto acontecer, por exemplo, o facto de algumas amostras apresentarem mais informação musical que outras(amostras com trechos de música mais exemplificativos) mesmo que mais curtas.

No caso das demonstrações com/sem presença de *noise* os resultados eventualmente podem fugir do esperado uma vez que, nos ruídos usados, há uns mais barulhentos e ruidosos que outros(mesmo com uma tentativa de contornar este facto).

Percentagens de participação

Fernando Lopes n.º 106358 - 33%

Maria João Sousa n.º 109488 - 33%

João Carvalho n.º 89059 - 33%

Recursos utilizados

[1]Programa getMaxFreqs, disponibilizado pelos professores
[2]Ficheiros .wav disponibilizados pelos professores
[3]http://sox.sourceforge.net/
[4]https://docs.python.org/3/library/gzip.html
[5]https://docs.python.org/3/library/bz2.html
[6]https://docs.python.org/3/library/lzma.html
[7]https://docs.python.org/3/library/zlib.htm