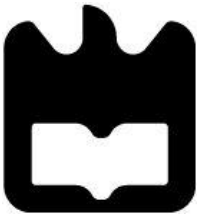
TAI - Assignment III

Universidade de Aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Mestrado em Engenharia Informática



Diogo Carvalho nº92969

Diogo Cunha nº95278

Rafael Baptista nº93367

31 de Janeiro de 2022

Índice

1. Introdução

O presente relatório pretende descrever o trabalho realizado na execução do terceiro projeto proposto na unidade curricular de Teoria Algorítmica da Informação.

Neste projeto, fomos desafiados a desenvolver e testar uma aplicação capaz de identificar automaticamente músicas. As músicas devem encontrar-se numa versão completa na base de dados que foi criada previamente, e a identificação deve ser efetuada com base em amostras de curta duração destas mesmas músicas, por exemplo, amostras de 10 segundos.

A identificação das músicas deve ser feita através do cálculo do valor de Normalized Compression Distance (NCD) entre cada par de música presente na base de dados e a amostra que se pretende identificar.

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

onde C(x) é o número de bits necessário pelo compressor C para representar x e C(x,y) é o número de bits necessário para representar x e y em conjunto.

Quanto menor o valor de NCD, maior é a similaridade entre a amostra que se pretende classificar e a música que se encontra na base de dados. Posto isto, após se calcular o valor de NCD para cada par composto pela amostra e cada uma das músicas presentes na base de dados, a amostra irá ser atribuída à música que pertence ao par com menor valor de NCD.

Por forma a que esta abordagem tenha sucesso, é necessário numa fase inicial converter cada música numa representação mais adequada para compressores. A esta representação mais adequada iremos dar o nome de assinatura. Esta assinatura é construída através da transformação do ficheiro de áudio num conjunto com as frequências mais significativas desse mesmo áudio. Isto é possível de ser obtido através da segmentação do ficheiro de áudio em partes e da computação das frequências mais significativas em cada parte segmentada. Esta assinatura será utilizada no cálculo do valor de NCD.

Finalmente, foi solicitado que fizéssemos um conjunto de diversas experiências por forma a compreender como diferentes fatores afetam a precisão da nossa aplicação. Segue-se uma lista com algumas das experiências que foram efetuadas:

- Utilização de diferentes compressores no cálculo do valor NCD.
- Adição/Remoção de ruído na amostra que se pretende classificar.
- Variação do volume do ruído introduzido na amostra.
- Variação do tipo de ruído introduzido na amostra.

2. Metodologia

Para a realização deste projeto, utilizámos a plataforma GitHub como repositório de código compartilhado. A estrutura do nosso repositório é constituída por uma área com o código desenvolvido e uma secção com o relatório escrito. Dentro da área de código, existe uma pasta (getmax) que contém o programa fornecido pelos professores para a criação de uma assinatura a partir de um ficheiro de áudio e ainda um conjunto de pastas relacionadas com o armazenamento das músicas e amostras. Estas pastas contêm a seguinte estrutura interna:

- Pasta “database”: Armazena todas as músicas pertencentes à base de dados e que já sofreram pré-processamento por forma a converter o formato e frequência de amostragem da música em formatos e frequências suportadas.
- Pasta “newMusics”: Deve ser utilizada para colocar todas as músicas novas que pretendemos introduzir na base de dados.
- Pasta “newVideos”: Deve ser utilizada para colocar todos os vídeos de áudio novos que pretendemos processar para posteriormente criar samples.
- Pasta “videoToSample”: Armazena todos os vídeos já processados a partir dos quais o utilizador pode criar samples para avaliar.
- Pasta “samples”: Armazena todas as amostras de músicas.
- Pasta “freqs”: Armazena todos os ficheiros de assinaturas que já foram criados anteriormente. Permite evitar a criação repetida de assinaturas das músicas da base de dados sempre que o programa é executado de novo.

Esta estrutura geral foi criada tendo por base a estrutura que foi recomendada pelos professores.

No que diz respeito à comunicação entre os elementos do grupo, esta foi efetuada através da plataforma Discord, onde foram discutidos os pontos críticos do nosso projeto, facilitando assim a cooperação à distância na realização do mesmo.

Relativamente ao código escrito, o mesmo foi desenvolvido na linguagem de programação Python. A estrutura do código é composta por:

.

- Programa *preprocesser.py* que é utilizado para efetuar o pré-processamento das novas músicas que foram colocadas na pasta “newMusics” e dos novos vídeos que são colocados na pasta “newVideos”. À medida que efetua o processamento dos ficheiros, o programa remove estes ficheiros da pasta origem e coloca os ficheiros processados na pasta destino (pasta “database” no caso das músicas e pasta “videoToSample” no caso dos vídeos para análise), automaticamente.
- Programa *sampleCreation.py* permite a criação de amostras de músicas com base num vídeo que se encontre na pasta “videoToSample”. Permite selecionar a duração da amostra, adicionar ruído à amostra, adaptar o volume de ruído e ainda escolher o tipo de ruído introduzido.
- Programa *musicfinder.py* que efetua a previsão da música a que uma determinada amostra pertence, com base numa base de dados e através do cálculo do valor de NCD.

3. Implementação

Nesta secção do nosso relatório iremos descrever a implementação desenvolvida para cada um dos nossos programas bem como analisar e justificar as principais decisões tomadas durante esta mesma fase de implementação.

3.1. preprocesser.py

Vamos começar por descrever a implementação do programa *preprocesser.py* visto que é responsável pelo pré-processamento das novas músicas e vídeos que pretendemos adicionar à nossa base de dados e pasta de vídeos para criação de amostras, respetivamente. O principal objetivo ao desenvolver este programa foi fornecer ao utilizador a capacidade de introduzir de forma facilitada as músicas de que gosta na base de dados do programa e os vídeos dos quais pretende criar amostras para classificar na pasta "videoToSample", sendo o programa responsável por processar esses ficheiros, enquanto ao mesmo tempo o programa pode também ser utilizado pelos programas *musicfinder.py* e *sampleCreation.py* para efetuar processamento inicial de ficheiros.

Como referido acima, este programa vai ser utilizado pelos programas *musicfinder.py* e *sampleCreation.py* pelo que foi desenhado para que possa ser utilizado de forma separada, ou seja, executado através da linha de comandos com o objetivo de simplesmente atualizar a base de dados e o banco de vídeos para criação de samples, ou possa ser utilizado pelos dois programas referidos para efetuar o processamento inicial de ficheiros.

Para correr este programa não é necessário especificar nenhum argumento.

Ao executar o programa, o primeiro passo a ser efetuado é a definição de algumas variáveis iniciais tais como:

- Lista de ficheiros para processar.
- Formatos de áudio suportados pelo programa.

De seguida, é iniciado o processo de pré-processamento destes ficheiros. Cada ficheiro é pré-processado de forma isolada, e é sempre indicado ao utilizador através do terminal o estado do programa através da atualização do número de ficheiros já processados.

No processamento de cada ficheiro, começamos por verificar se o ficheiro contém um formato suportado. A lista de formatos suportados é a seguinte:

- .wav
- .flac
- .mp3
- .mp4

Se o ficheiro tiver um formato não suportado, o utilizador é informado e aquele ficheiro em específico é ignorado. Se o formato for suportado, no caso de o mesmo ser diferente de .wav, o ficheiro é convertido para o formato .wav. De seguida, é alterada a taxa de amostragem do ficheiro para o valor de 44100 hertz que é o valor padrão para ficheiro áudio CD. Este novo ficheiro já processado é colocado na localização de destino que, no caso do

programa *musicfinder.py* é a pasta “database” e no caso do programa *sampleCreation.py* é a pasta “videoToSample”. No fim do processo, o ficheiro inicial por processar é removido da pasta origem para evitar que volte a ser processado numa execução futura do programa.

3.2. sampleCreation.py

Prosseguindo para o programa *sampleCreation.py*, o mesmo tem como principal objetivo, mais uma vez, oferecer ao utilizador ferramentas que lhe permitam utilizar a nossa plataforma da forma que lhe fornecer maior proveito. Posto isto, decidimos criar um programa que permite a construção de amostras de músicas com base numa música que pertença a um ficheiro presente na pasta “videoToSample”. O utilizador tem a capacidade de decidir qual a duração e segundo de início da amostra, se pretende adicionar ruído ou não, o volume de ruído que pretende adicionar e ainda o tipo de ruído.

Posto isto, para correr o programa *sampleCreation.py*, são necessários especificar 4 argumentos:

- *-ftarget*: Define a música presente na base de dados da qual se pretende criar a amostra. Deve conter o nome completo do ficheiro da música à exceção do formato do ficheiro. Por exemplo, para criar amostra da música contida no ficheiro “Novo Hit do meu Cantor Favorito.wav”, o parâmetro deve ter o valor de “Novo Hit do meu Cantor Favorito”.
- *-start*: Define o momento onde o utilizador pretende que a amostra inicie. Em caso de não ser fornecido pelo utilizador, terá um valor por defeito de 0 segundos.
- *-duration*: Define a duração da amostra que se pretende criar. Em caso de não ser fornecido pelo utilizador, terá um valor por defeito de 20 segundos.
- *-noise*: Define se o utilizador pretende ou não adicionar ruído à amostra e o volume desse mesmo ruído. Em caso de assumir valor de 0, a amostra não terá ruído. Por defeito possui o valor de 0. O valor deve encontrar-se no intervalo [0, 1].
- *-type*: Define o tipo de ruído que o utilizador pretende introduzir na amostra.

Inicialmente, o primeiro passo a ser efetuado é a verificação dos parâmetros fornecidos pelo utilizador. No que diz respeito ao parâmetro *ftarget*, existe a verificação de se o ficheiro existe na base de dados. Caso o ficheiro não exista, o utilizador é informado e o programa encerra. Os parâmetros *start* e *duration* devem de ser números inteiros e superiores a 0, visto que o objetivo dos mesmos são definir o tempo de início e duração da amostra. Por sua vez, o parâmetro *noise* é verificado que é um float e que o valor se encontra no intervalo [0, 1], sendo 0 igual a não introduzir ruído algum na amostra e 1 a introdução do ruído máximo. Finalmente, para o parâmetro *type* é verificado se o tipo de ruído selecionado é suportado ou não. Segue-se a lista de tipos de ruídos suportados:

- whitenoise
- pinknoise
- brownnoise

De seguida, existe a definição de algumas variáveis iniciais, tais como os caminhos para as pastas “newVideos”, “videoToSample” e “samples”. Em seguida é executado o

programa preprocesser.py por forma a processar qualquer novo vídeo que se encontre na pasta “newVideos”. Em sequência, é criada uma amostra que inicia e possuí o tamanho definido pelo utilizador a partir do ficheiro selecionado pelo mesmo também.

Caso o utilizador tenha selecionado um valor de ruído igual a 0, ou seja, tenha selecionado não introduzir qualquer tipo de ruído à amostra, o programa notifica o utilizador de que a amostra foi criada com sucesso e encerra. Caso contrário, o programa vai utilizar a amostra já criada para introduzir o volume de ruído, bem como o tipo de ruído selecionado pelo utilizador e só depois encerrar.

3.3. musicfinder.py

Relativamente ao principal programa desenvolvido no nosso projeto (*musicfinder.py*), o mesmo tem como objetivo principal conseguir identificar a que música pertence uma determinada amostra de curta duração.

Posto isto, para correr o programa *musicfinder.py*, são necessários especificar 2 argumentos:

- *-ftarget*: Define a amostra que se pretende avaliar e identificar a que música pertence. Deve conter o nome completo do ficheiro da amostra à exceção do formato do ficheiro. Por exemplo, para analisar a amostra contida no ficheiro “Amostra do Novo Hit do meu Cantor Favorito.wav”, o parâmetro deve ter o valor de “Amostra do Novo Hit do meu Cantor Favorito”.
- *-compressor*: Define o compressor que o utilizador pretende que seja utilizado durante os cálculos dos valores de NCD.
- *-top*: Define o número de resultados, ou seja, músicas que o utilizador pretende que o programa retorne.

Inicialmente e de forma idêntica ao programa anterior, o primeiro passo a ser efetuado é a verificação dos parâmetros fornecidos pelo utilizador. No que diz respeito ao parâmetro *ftarget*, existe a verificação de se o ficheiro existe na pasta “samples”. Caso o ficheiro não exista, o utilizador é informado e o programa encerra. No que diz respeito ao parâmetro *compressor,* é verificado se o mesmo pertence ao conjunto de compressores suportados pela aplicação. Segue-se a lista de compressores suportados:

- gzip.
- lzma.
- bz2/bzip2 (ambas as nomenclaturas são aceites).
- zlib.
- lz4.

No seguimento, o programa começa por inicializar algumas variáveis iniciais como os caminhos para as pastas “newMusics”, “database” e “samples” e após isso efetua uma chamada ao programa *preprocesser.py* descrito anteriormente para verificar se existe alguma atualização da base de dados pendente, ou seja, se existe alguma música na pasta “newMusics” à espera de ser processada e adicionada.

De seguida, o programa vai buscar à base de dados as músicas que pertencem à mesma e para cada uma delas verifica se já existe uma assinatura da mesma criada na pasta

“freqs/database/”. Caso não exista, existe uma chamada ao programa “GetMaxFreqs” fornecido pelos professores que constrói a assinatura do ficheiro de áudio que necessitamos.

No fim de criar as assinaturas para todas as músicas da base de dados, é também criada a assinatura da amostra selecionada pelo utilizador e que pretendemos classificar.

Após a criação das assinaturas, inicia-se o processo de cálculo dos valores de NCD. Para isso, o primeiro passo é ler a assinatura da amostra e efetuar uma chamada à função *getNumBitsFromCompressor()* que recebe o conteúdo da assinatura e o compressor selecionado pelo utilizador e retorna o número de bits necessários para comprimir a assinatura utilizando o compressor selecionado. Este valor é armazenado e posteriormente começa um ciclo de iteração sobre todas as músicas pertencentes à base de dados, onde para cada música *x,* vamos efetuar a leitura da assinatura associada a *x*, obter o número de bits necessário para comprimir a assinatura com o compressor desejado, concatenar o conteúdo de ambas as assinaturas, isto é, da assinatura da amostra e da assinatura de *x*, obter o número de bits necessário para comprimir este conteúdo concatenado e no fim calcular o NCD(amostra, *x*). Cada valor calculado é armazenado num dicionário designado *ncd_dicts* cujas chaves são os nomes das músicas e o valor associado a cada chave é o valor do NCD para a música em questão.

Para concluir, o dicionário é ordenado pelo valor de NCD e é representada uma lista com o número de músicas selecionadas pelo utilizador, músicas estas que possuem os menores valores de NCD calculados.

4. Análise de resultados

Nesta secção do relatório iremos apresentar algumas experiências efetuadas com os programas desenvolvidos e analisar os resultados obtidos a partir das mesmas.

Para realização destes testes foram recolhidas um total de 29 músicas, que incluíam ficheiros áudio com formato .wav, .flac, .mp3 e .mp4.

4.1. Variação NCD em função do compressor

Inicialmente foi efetuado um teste cujo objetivo era compreender se o valor de NCD entre uma amostra e a música completa da qual a amostra foi criada variava com a utilização de diferentes compressores. Para isso, foi selecionada uma música aleatória da nossa base de dados, criada uma amostra através do programa *sampleCreation.py* com 20 segundos e sem qualquer ruído a partir da música selecionada. Foram testados todos os compressores suportados na nossa aplicação que já foram enumerados acima.

A música escolhida foi "ZAYN, Sia - Dusk Till Dawn".

Os valores numéricos de NCD obtidos, bem como a posição em que a música se encontrava na previsão feita, encontram-se na tabela abaixo:

Compressor	***gzip***	***lzma***	***bzip2***	***zlib***	***lz4***
NCD	0.91444	0.89482	0.94541	0.91493	0.90616
Posição	1º	1º	1º	1º	1º

Tabela 1: Variação dos valores de NCD em função do compressor utilizado

A partir dos resultados demonstrados na tabela acima, podemos verificar que efetivamente existe uma variação do valor de NCD calculado com diferentes compressores. Isto deve-se ao facto de que, obviamente, os compressores têm diferentes implementações pelo que alguns deles conseguem comprimir de forma mais eficiente as assinaturas como é o caso do lzma. É importante também realçar que, como podemos ver na tabela, para todos os compressores a previsão efetuada pelo programa é a correta, ou seja, o NCD calculado entre o par formado pela amostra e a música completa foi o NCD mais baixo de entre todos os calculados.

Com este resultado o programa revelou-se bastante promissor na deteção das músicas a que as amostras pertencem.

4.2. Variação NCD em função do tempo da amostra

Visto que os resultados do teste anterior demonstraram que a escolha do compressor tem influência no valor de NCD calculado, procuramos detetar que outros fatores podem também influenciar este valor.

Tendo isto em consideração, o próximo teste que realizamos teve como objetivo verificar a influência do tempo da amostra no valor de NCD. O teste consistiu na escolha de uma música aleatória, criação de amostras com diferentes valores de duração e sem qualquer ruído associado a partir da música completa selecionada e verificação dos valores de NCD obtidos entre a música completa e as amostras criadas. Este processo foi efetuado para todos os compressores suportados.

A música escolhida foi “Adele - Easy On Me”.

Os valores numéricos de NCD obtidos, bem como a posição em que a música se encontrava na previsão feita, encontram-se na tabela abaixo:

	gzip		***lzma***		***bzip2***		***zlib***		***lz4***	
Duração	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***
10s	0.96842	1º	0.95679	1º	0.97628	1º	0.96895	1º	0.96097	1º
15s	0.94957	1º	0.93519	1º	0.96259	1º	0.95009	1º	0.94159	1º
20s	0.93168	1º	0.91690	1º	0.94928	1º	0.93224	1º	0.92017	1º
25s	0.90824	1º	0.89079	1º	0.93554	1º	0.90878	1º	0.89529	1º
30s	0.88621	1º	0.86871	1º	0.92042	1º	0.88673	1º	0.87294	1º
40s	0.83997	1º	0.82028	1º	0.89028	1º	0.84041	1º	0.82697	1º
50s	0.78848	1º	0.76614	1º	0.85544	1º	0.78878	1º	0.77843	1º
60s	0.74295	1º	0.72270	1º	0.81993	1º	0.74322	1º	0.73193	1º

Tabela 2: Variação dos valores de NCD em função do tempo da amostra utilizada

Através da análise dos valores presentes na tabela anterior podemos verificar que o tempo de duração da amostra influencia o valor de NCD calculado. Relativamente a esta influência podemos concluir que existe uma correlação negativa entre o tempo da amostra e o valor de NCD. À medida que o tempo da amostra aumenta, o valor de NCD diminui, o que quer dizer que a amostra e a música têm uma maior similaridade, o que é natural visto que a amostra analisada se aproxima da música completa.

Adicionalmente podemos ainda verificar que para todas as previsões efetuadas acima, em todas elas a música a partir da qual a amostra foi construída foi dada como sendo a primeira da lista, ou seja, a que possuiu menor valor de NCD. Estes resultados aumentaram ainda mais a nossa perceção de que a aplicação poderia ter uma boa precisão. No entanto estes valores de precisão ainda serão avaliados mais à frente.

4.3. Variação NCD em função do volume de ruído adicionado

Prosseguindo com os testes realizados, após verificarmos a influência do tipo de compressor e do tempo da amostra utilizado, procuramos descobrir se existe uma ligação entre o volume de ruído adicionado à amostra e o valor de NCD calculado entre a amostra e a música completa a que a amostra corresponde.

Para tal efeito, foi efetuada a escolha de uma música aleatória a partir da qual se criou um conjunto de amostras com diferentes valores de ruído adicionado. Todas as amostras possuíam a mesma duração. Este processo foi efetuado para todos os compressores suportados.

A música escolhida foi “Ed Sheeran - Bad Habits” e a duração de todas as amostras foi de 20 segundos. O tipo de ruído adicionado foi “whitenoise” para todas as amostras. Os valores numéricos de NCD obtidos, bem como a posição em que a música se encontrava na previsão feita, encontram-se na tabela abaixo:

	gzip		***lzma***		***bzip2***		***zlib***		***lz4***	
Volume Ruído	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***
0.05	0.92857	1º	0.90889	1º	0.94829	1º	0.92911	1º	0.92233	1º
0.10	0.93868	1º	0.91432	1º	0.95297	1º	0.93776	1º	0.93114	1º
0.15	0.94938	1º	0.92408	1º	0.95925	1º	0.94855	1º	0.94129	1º
0.20	0.95511	1º	0.93015	1º	0.96188	1º	0.95743	1º	0.95074	1º
0.30	0.96485	1º	0.93731	1º	0.96661	1º	0.96558	1º	0.95966	1º
0.50	0.97492	1º	0.95054	1º	0.97405	1º	0.98038	1º	0.97263	1º
0.70	0.98671	1º	0.95879	1º	0.97784	1º	0.98611	1º	0.97713	1º
1.00	0.99558	1º	0.96746	1º	0.98680	1º	0.99513	1º	0.98741	1º

Tabela 3: Variação dos valores de NCD em função do volume de ruído da amostra utilizada

A observação dos dados apresentados na tabela acima permite delinear algumas conclusões importantes. A primeira conclusão a retirar é que, como esperado, o volume de ruído introduzido na amostra influencia o valor de NCD calculado. Quanto maior o ruído, maior o valor de NCD, menor a semelhança entre a amostra e a música completa associada à amostra.

Adicionalmente, a segunda conclusão a retirar é que mais uma vez a aplicação teve um comportamento excelente, acertando todas as previsões. Em todos os casos, a música prevista foi a correta. Obviamente estes resultados revelaram-se satisfatórios

4.4. Variação NCD em função do tipo de ruído adicionado

Seguindo a mesma linha de pensamento que nos tem guiado até aqui, ao observarmos que o volume de ruído também influencia o valor de NCD calculado, procuramos descobrir se o tipo de ruído introduzido na amostra tem também alguma influência nos resultados obtidos. Posto isto, para este teste foi efetuada a escolha de uma música aleatória a partir da qual se criou um conjunto de amostras com diferentes tipos de ruído adicionado. Para cada tipo de ruído, foram testadas amostras com diferentes valores de volume de ruído adicionado. Todas as amostras possuíam a mesma duração. Este processo foi efetuado para todos os compressores suportados.

A música escolhida foi “Jon Bellion - All Time Low (Official Music Video)” e a duração de todas as amostras foi de 20 segundos. Os valores numéricos de NCD obtidos, bem como a posição em que a música se encontrava na previsão feita, encontram-se na tabela abaixo:

		gzip		***lzma***		***bzip2***		***zlib***		***lz4***	
Tipo de Ruído	***Volume***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***	***NCD***	***Pos***
Whitenoise	***0.2***	0.96338	1º	0.93576	1º	0.96596	1º	0.96379	1º	0.95786	1º
	0.5	0.98173	1º	0.95589	1º	0.97657	1º	0.98219	1º	0.97676	1º
	0.8	0.99223	1º	0.96768	1º	0.98189	1º	0.99273	1º	0.98644	1º
Pinknoise	***0.2***	0.96423	1º	0.94084	1º	0.96989	1º	0.96468	1º	0.95925	1º
	0.5	0.99078	1º	0.96930	1º	0.98820	1º	0.99129	1º	0.98380	1º
	0.8	0.99949	1º	0.98292	1º	0.99959	1º	1.00000	1º	0.99311	1º
Brownnoise	***0.2***	0.99762	1º	0.97804	1º	0.99742	1º	0.99813	1º	0.98936	1º
	0.5	1.00612	3º	0.98658	6º	1.00528	1º	1.00663	3º	0.99683	1º
	0.8	1.00743	9º	0.99004	6º	1.00623	2º	1.00799	8º	0.99851	6º

Tabela 4: Variação dos valores de NCD em função do tipo de ruído da amostra utilizada

Os números da tabela anterior permitem concluir que também o tipo de ruído influencia o valor de NCD, e consequentemente, a previsão efetuada. Como podemos verificar, o valor de NCD é mais elevado para o tipo brownnoise enquanto assume valores menores para o tipo whitenoise. Isto deve-se ao facto de diferentes tipos de ruído introduzirem diferentes valores de frequência nas músicas.

Para além disto, podemos também concluir que efetivamente a nossa aplicação teve um desempenho excelente para o ruído whitenoise e pinknoise, onde efetuou todas as previsões de forma correta com todos os compressores suportados. Já para o brownnoise,

os compressores revelaram possuir uma maior dificuldade para este tipo específico de ruído tendo alguns deles demonstrado conseguir efetuar melhores previsões do que outros.

De uma forma geral consideramos que os resultados demonstram uma performance satisfatória da aplicação e que a mesma consegue ter um alto nível de robustez contra vários tipos de ruído e com diferentes valores de volume de ruído.

4.5. Precisão da aplicação para amostras sem ruído

Nesta fase do nosso processo de testagem já conseguimos identificar diversos fatores que influenciam o valor de NCD calculado e que, por isso, possuem influência nas previsões que a aplicação executa. Posto isto, pretendemos agora avaliar a performance da nossa aplicação através da identificação do número de previsões corretas e erradas que são efetuadas.

Para uma primeira fase do teste, para cada música presente na base de dados (29 músicas) foram criadas três amostras com duração de 10, 20 e 30 segundos respetivamente e sem qualquer ruído associado. Para cada amostra foi posteriormente efetuada a previsão da música a que pertence utilizando cada um dos compressores suportados.

Os resultados obtidos encontram-se descritos na tabela abaixo:

	gzip	***lzma***	***bzip2***	***zlib***	***lz4***
Duração	***Acertos***	***Acertos***	***Acertos***	***Acertos***	***Acertos***
10s	29/29	29/29	29/29	29/29	29/29
20s	29/29	29/29	29/29	29/29	29/29
30s	29/29	29/29	29/29	29/29	29/29

Tabela 5: Número de previsões acertadas para amostras sem ruído das músicas presentes na base de dados

Como já estávamos à espera com os resultados que vínhamos a obter na realização das experiências anteriores, a aplicação efetuou todas as previsões para cada compressor e para cada música da base de dados de forma correta. Não houve qualquer erro. Consideramos que estes resultados são bastante positivos.

4.6. Precisão da aplicação para amostras com ruído

Visto que os resultados obtidos na secção anterior foram extremamente positivos, procurámos avaliar também a performance da aplicação para amostras em que foi adicionado ruído durante a sua criação. Assim sendo, neste teste foram criadas para cada música presente na base de dados nove amostras. Cada conjunto de nove amostras continha três para o tipo de ruído whitenoise, três para pinknoise e três de brownnoise. Em cada um destes conjuntos de três amostras, existia uma amostra com volume de ruído 0.2, outra com volume de ruído igual a 0.5 e a última possuía um volume igual a 0.8. O tempo de cada amostra foi fixo e assumia o valor de 20 segundos. Para cada amostra foi posteriormente efetuada a previsão da música a que pertence utilizando cada um dos compressores suportados.

Os resultados obtidos encontram-se descritos na tabela abaixo:

		gzip	***lzma***	***bzip2***	***zlib***	***lz4***
Tipo de Ruído	***Volume***	***Acertos***	***Acertos***	***Acertos***	***Acertos***	***Acertos***
Whitenoise	***0.2***	28/29	29/29	29/29	28/29	29/29
	0.5	27/29	27/29	27/29	27/29	28/29
	0.8	23/29	26/29	26/29	23/29	27/29
Pinknoise	***0.2***	28/29	28/29	28/29	28/29	28/29
	0.5	23/29	26/29	27/29	21/29	27/29
	0.8	11/29	18/29	20/29	7/29	19/29
Brownnoise	***0.2***	18/29	22/29	21/29	17/29	23/29
	0.5	2/29	4/29	4/29	2/29	3/29
	0.8	2/29	4/29	0/29	2/29	1/29

Tabela 6: Número de previsões acertadas para amostras com ruído das músicas presentes na base de dados

Pela análise dos dados presentes na tabela anterior podemos verificar que efetivamente o tipo de ruído influência o número de previsões acertadas pelo programa.

A aplicação consegue lidar bem com o ruído do tipo whitenoise, mesmo com valores de volume elevados. No que diz respeito ao pinknoise, o programa ainda consegue ter uma boa taxa de acerto para valores de ruído baixo e médio. Contudo, para o brownnoise a aplicação consegue ter uma boa taxa de acertos apenas quando o volume de ruído é baixo.

Apesar de tudo, nas ocasiões em que a previsão erra, normalmente a música correta revelou estar entre as melhores músicas, pelo que se o utilizador escolher apresentar uma lista de 3, 5 ou 10 músicas invés de apenas a previsão principal, poderá ainda encontrar a música que pretende.

4.7. Tempo de execução

Por fim, procurámos avaliar a rapidez de execução do programa *musicfinder.py* através da medição do seu tempo de execução. De relembrar que estes testes foram efetuados com uma base de dados composta por 29 músicas distintas, pelo que em cada execução do programa foi efetuado o cálculo do valor de NCD para 29 pares de ficheiros áudio. Em todos os testes não foram colocadas quaisquer músicas novas na pasta "newMusics" para evitar que fosse contabilizado o tempo de execução do programa *preprocesser.py* no processamento destas novas músicas.

Os tempos foram medidos na análise de uma amostra de 20 segundos criada a partir da música "T-Rex - Duvidava (Video Oficial)" para cada um dos diferentes tipos de compressores suportados e encontram-se presentes na tabela abaixo:

	gzip	***lzma***	***bzip2***	***zlib***	***lz4***
Tempo de Execução (s)	*0.129*	*0.674*	*0.285*	*0.126*	*0.040*

5. Conclusão

A realização deste projeto permitiu-nos efetuar uma aplicação prática dos conhecimentos que fomos adquirindo nas aulas teóricas relativos à complexidade de Kolmogrov através da criação de um programa que associa amostras de curta duração de músicas à música completa da qual a amostra foi retirada, desde que esta se encontre na nossa base de dados.

O desenvolvimento deste programa foi baseado no cálculo do valor de Normalized Compression Distance (NCD) que permitiu detetar similaridade entre assinaturas de cada música utilizada. A criação destas assinaturas foi possível através de um código fornecido pelos professores onde permitiu-nos compreender como é possível representar um ficheiro de áudio complexo através de um conjunto de frequências desse mesmo áudio.

Adicionalmente, o desenvolvimento desta aplicação permitiu-nos trabalhar com ferramentas novas na realização de tarefas que também eram novas para nós, tais como a manipulação de ficheiros de áudio através da conversão de formato, taxa de amostragem e ainda todo o processo de adição e remoção de ruído.

Por fim, consideramos que este projeto foi útil também para o desenvolvimento das nossas capacidades individuais como futuros engenheiros informáticos, uma vez que se tratou de um desafio novo que conseguimos, no nosso ponto de vista, realizar com sucesso.