TAI 2020-21

Projeto 3

Tema: **Shazam baseado em Compressão**
Autores: **Mário Silva nºmec 93430, Daniel Gomes nºmec 93015, João Magalhães nºmec 79923**
Date: **28/12/2021**

Index

1 Introdução

Este projeto foi desenvolvido no contexto da disciplina de *Teoría Algorítimica da Informação* da *Universidade de Aveiro*. Consiste no uso da *Normalized Compression Distance* (NCD), que é uma aproximação usada na compressão da *Non-Computable Normalized Information Distance* (NID), que representa o número de bits necessários para efetuar a compressão entre duas *strings x* e *y*. Assim, um valor desta distância perto de 0 correlaciona-se com uma similaridade entre ambas as *strings*, enquanto um valor perto de 1 será sinónimo de uma dissimilaridade.

2 Objetivos

Tendo em conta o que foi referido anteriormente, os objetivos deste trabalho consistem no desenvolvimento de um programa que permitisse a identificação automática de músicas, como o ”Shazam”, mas através da NCD, *Normalized Compressed Distance*. Consequentemente, neste processo deverão ser utilizadas pequenas amostras como, por exemplo, 5 segundos, a serem pesquisadas numa base de dados. Esta foi construída pelos elementos do grupo, agregando um conjunto de músicas diversas.

Para que esta abordagem fosse bem-sucedida foi necessário transformar os ficheiros de áudio para uma representação das frequências mais significativas destes, de forma a serem utilizadas adequadamente pelos compressores à qual recorremos.

Além disto, outro objetivo é também a adição de ruído na amostra para testar a robustez da estratégia aplicada para a identificação das músicas.

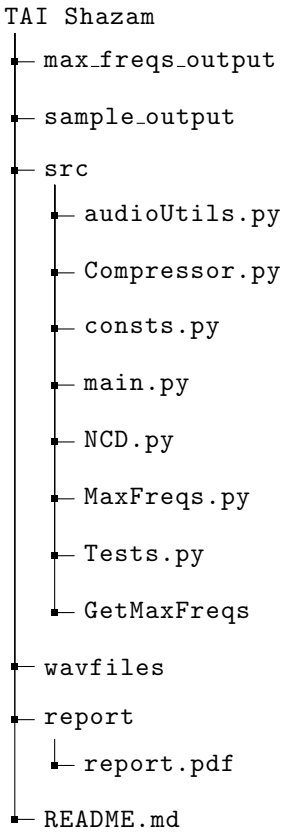
3 Implementação

Nesta secção abordaremos como realizámos a nossa implementação, que organização utilizamos, os módulos e as estruturas de dados desenvolvidas, bem como a estratégia utilizada.

3.1 Organização

Na figura abaixo é possível visualizar toda a organização do trabalho desenvolvido. Na pasta ***src***, encontram-se presente os módulos de código criados (com exceção do binário *GetMaxFreqs*, enquanto que no diretório ***wav_files*** estão presentes ficheiros de áudio que irão ter o papel de ”base de dados” de músicas.

Tanto o diretório *max_freqs_ouput* como o *sample_output*, têm a função de armazenar os ficheiros da base de dados e o ficheiro de áudio a ser identificado normalizados, respetivamente. Além disto, o documento ***README.md*** apresenta as instruções de como correr os programas elaborados. Finalmente, o ficheiro ***report.pdf*** representa este documento, o relatório de trabalho produzido.



3.2 Módulos Desenvolvidos

3.2.1 MaxFreqs

Este módulo tem como objetivo, calcular as frequências dos ficheiros de áudio e escrevê-las para outro ficheiro, recorrendo ao programa *GetMaxFreqs*. Estes ficheiros com as frequências são depois utilizados mais à frente.

Ao ser inicializado um objeto desta classe, é calculada as frequências para todos os ficheiros de referência, se estas não estiverem presentes no diretório das frequências. Através do mesmo método, é possível aplicar o mesmo processo, mas apenas para um ficheiro de áudio, que será o ficheiro alvo.

3.2.2 audioUtils

O módulo *audioUtils* funciona como um *wrapper*, visto que fornece métodos que abstraem a execução de dois comandos no Sistema Operativo. No primeiro, *add_noise*, através do software *Sox*, corre-se um comando que permite a adição de *noise* a um ficheiro de áudio. Já no segundo, tem-se como objetivo correr um comando do *sox* que permitirá efetuar o ”corte” de um ficheiro de áudio num intervalo de tempo específico.

3.2.3 Compressor

Este módulo serve como uma camada de abstração para os diferentes tipos de compressores. É um módulo bastante simples, apenas faz o *import* ao tipo de compressor fornecido como argumento. Os compressores disponíveis, são o *gzip*, *bzip2* e *lzma*.

Possui um método *compress(data)*, que executa a compressão dos dados passados como argumento, e retorna o tamanho dessa compressão, ou seja, retorna o número de *bytes* necessários para a compressão desses dados.

3.2.4 NCD

O módulo *NCD* contém toda a lógica para efetuar a identificação do ficheiro de áudio requerido pelo utilizador do programa. Para tal, recorre-se ao cálculo da *Normalized Compressed Distance* (NCD), cuja fórmula encontra-se representada abaixo.

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

Em que C(x) traduz-se no número de bits necessários para representar o segmento de música do ficheiro alvo, e C(y) o número de bits necessários para representar o ficheiro de áudio de referência da base de dados. Assim, podemos assumir que C(x,y) espelha o número de bits necessários para efetuar a representação das duas juntas.

Esta classe recebe como argumento um compressor a utilizar no cálculo acima descrito, escolhido pelo utilizador.

Possui também um método *recognize_music()*, que, tal como o seu nome indica, irá efetuar a identificação do ficheiro de áudio alvo. Neste processo recorre-se ao cálculo da *NCD* e, posteriormente, do número de bits necessários, entre o segmento do ficheiro de áudio alvo, e todos os ficheiros presentes no diretório da base de dados, armazenando os valores obtidos em cada combinação possível e o respetivo ficheiro de referência. Assim, o cálculo que obteve menor número de bits, irá conter o ficheiro de áudio identificado.

3.2.5 Main

Como é percetível pelo nome, este módulo representa o programa principal, onde é fornecido um leque de opções aos utilizadores, com vista ao que pretendam desempenhar no programa. A listagem abaixo representa todas as opções possíveis de realizar:

- *-s*, o nome do ficheiro alvo a ser identificado;
- *-m*, o nome de um diretório, com diversos ficheiros de referência a serem utilizados para identificar o ficheiro alvo;
- *-c*, o nome do compressor para ser utilizado no módulo *NCD*;
- *-n*, o nível de ruído a adicionar ao ficheiro alvo;
- *-t*, o instante de tempo para se efetuar o ”corte” do ficheiro de áudio a ser identificado
- *-d*, a duração do ”corte” do ficheiro de áudio a ser identificado.;
- *-r*, executar o módulo de testes;

Esta classe é também responsável pela efetuação das operações de acordo com os argumentos definidos. Assim, caso se escolha aplicar ruído, e/ou o ”corte” do ficheiro de música a ser identificado, estas operações são primeiramente realizadas. Posteriormente, procede-se à normalização tanto do alvo como dos ficheiros da base de dados, extraindo apenas as máximas frequências de cada um destes. Finalmente, como se encontram todas as pré-condições necessárias, realiza-se a operação de identificação do segmento de música alvo.

4 Resultados Obtidos

Nesta secção serão apresentados os resultados que obtivemos e testes efetuados com vista à validação da solução desenvolvida.

Desenvolvemos uma base de dados com 89 músicas. De notar, que como a execução de testes com uma base de dados deste tamanho demorará demasiado tempo com os ficheiros de música originais, decidimos apenas usar excertos de cada música.

De modo a testar o funcionamento e a qualidade da identificação das músicas do nosso programa, efetuamos um conjunto de testes. Todos estes testes foram executados a percorrer a base de dados das músicas todas até utilizar todos os ficheiros como música a identificar, e sempre usando todas como referência.

4.1 Testes com diferentes comprimentos de ”corte” aplicados

Com vista a testar o quão preciso é a funcionalidade de identificação do ficheiro de áudio alvo, que tenha sido submetido a um ”corte” do seu tamanho original procedeu-se à conduta de um teste onde cada uma da músicas foram sujeitas a um corte com tamanho final de 1, 3 e 5 segundos.

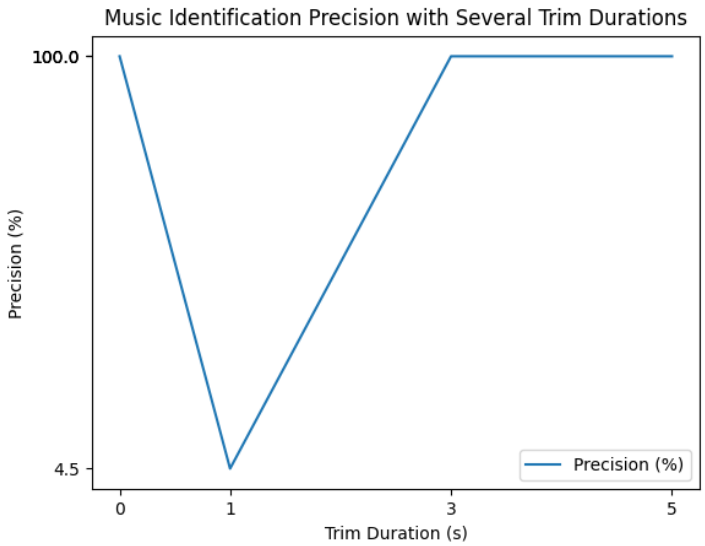


Figure 1: Variação da Precisão da identificação em função do comprimento do corte do segmento de música alvo

Como se pode observar pela figura 1 acima, para quase todos os valores de ”corte” mencionados anteriormente, obteve-se uma precisão de 100%, tendo sido o caso do corte para o tamanho final de 1 segundo a exceção, com cerca de 4.5% de precisão. Este valor acaba por ser expectável já que este intervalo de tempo é demasiado curto para ser comparado com os ficheiros originais da base de dados. De notar que na figura também é possível observar o valor de precisão para um corte de 0 segundos: este caso representa a situação de não ter sido aplicado qualquer ”corte” ao seu tamanho prévio.

4.2 Testes com diferentes níveis de ruído

De modo a testar a robustez do programa à adição de ruído, foram realizados testes para 4 níveis diferentes de ruído, 0, ou seja, sem ruído, que serve de comparação, 0.1, 0.5 e 1.

A figura 2 abaixo, demonstra os resultados obtidos. Pode-se observar que com uma adição baixa de ruído 0.1, a precisão permanece nos 100%, e à medida que se aumenta o ruído, a precisão tende a diminuir. No entanto, verifica-se que com um ruído algo elevado, 0.5, consegue-se obter uma relativamente boa precisão de, aproximadamente, 97%.

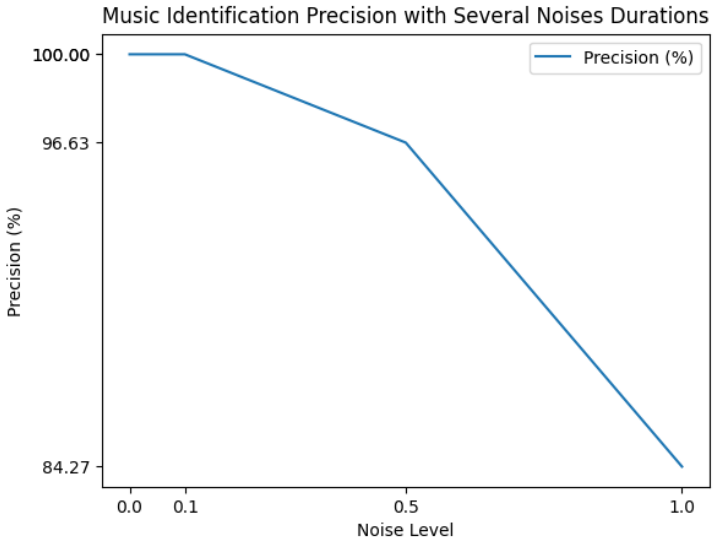


Figure 2: Variação da Precisão da identificação em função do nível de ruído aplicado no segmento de música alvo

4.3 Testes com diferentes compressores

Posteriormente também se procedeu à testagem dos diversos algoritmos de compressão disponíveis: gzip, bzip2 e lzma. Na figura seguinte encontra-se um gráfico que mostra a precisão da identificação de cada música em função do tipo de compressor escolhido. De notar, que não foi aplicado qualquer ”corte” ou ruído às músicas testadas.

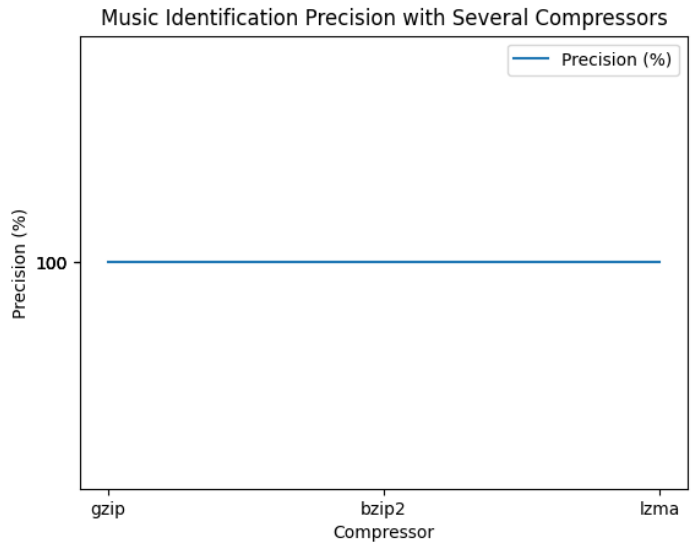


Figure 3: Variação da Precisão da identificação em função do algoritmo de compressão utilizado

Facilmente consegue-se perceber que a precisão foi máxima, independentemente do compressor. Contudo, no que toca a tempos de execução, todos estes são bastantes diferentes, algo que foi possível notar no teste seguinte: a variação do tempo de execução para identificar os ficheiros de música, para cada um dos compressores.

Pela figura abaixo 4, pode-se observar os tempos de execução com os diferentes compressores. O compressor *lzma* demora muito mais tempo que o *gzip* e o *bzip2*.

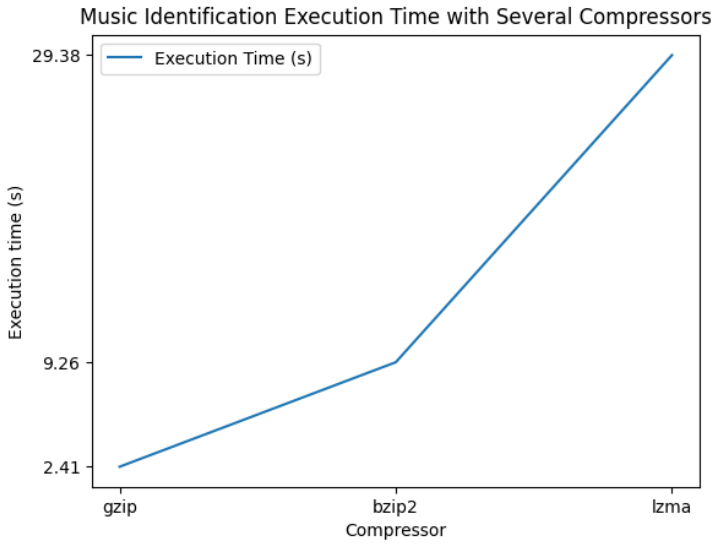


Figure 4: Variação do Tempo de Execução da identificação em função do algoritmo de compressão utilizado

4.4 Resultados gerais

De modo a obter uma visão global de como todos os fatores influenciam a tarefa de identificação de músicas, criamos um método que testava todas as combinações possíveis destes fatores.

Foram utilizados os mesmos conjuntos de valores para os fatores, 4 para o comprimento do ”corte” e para o nível de ruído, mas apenas 2 para o tipo de compressor, o compressor *lzma* não foi incluído no teste devido ao seu elevado tempo de execução.

No total foram realizados 32 combinações de fatores nos testes.

Trim Duration (s)	Noise Level	Compressor	Execution Time (s)	Precision
0	0	gzip	3.4	100.00%
0	0	bzip2	11.85	100.00%
0	0.1	gzip	2.91	100.00%
0	0.1	bzip2	15.11	100.00%
0	0.5	gzip	3.44	96.63%
0	0.5	bzip2	14	94.38%
0	1	gzip	3.43	84.27%
0	1	bzip2	12.82	83.15%
1	0	gzip	6.09	4.49%
1	0	bzip2	9.82	1.12%
1	0.1	gzip	5.72	1.12%
1	0.1	bzip2	15.01	1.12%
1	0.5	gzip	5.88	1.12%
1	0.5	bzip2	15.21	1.12%
1	1	gzip	7.53	1.12%
1	1	bzip2	14.96	1.12%
3	0	gzip	9.2	100.00%
3	0	bzip2	16.39	98.88%
3	0.1	gzip	5.02	97.75%
3	0.1	bzip2	14.46	89.89%
3	0.5	gzip	5.75	76.40%
3	0.5	bzip2	15.65	44.94%
3	1	gzip	6.3	47.19%
3	1	bzip2	15.83	22.47%
5	0	gzip	7.55	100.00%
5	0	bzip2	11.51	100.00%
5	0.1	gzip	5.66	98.88%
5	0.1	bzip2	15.79	98.88%
5	0.5	gzip	6.18	94.38%
5	0.5	bzip2	18.82	94.38%
5	1	gzip	7.19	76.40%
5	1	bzip2	16.96	77.53%

Table 1: Resultados Gerais de Identificação de Músicas

De uma forma geral, pelos resultados da tabela consegue-se averiguar que os valor mais reduzidos de precisão ocorreram em testes que envolveram um nível de ruído de 1 e/ou ”cortes” do ficheiro original para o tamanho de 1 segundo. Contudo noutros casos em que quer ruído quer o ”corte” foi aplicado às músicas testadas, obtiveram -se valor de precisão consideravelmente altos.

Além disto, o compressor *gzip* teve valores bastante próximos de precisão do compressor *bzip2*, tendo sido notada a maior diferença no teste com nível de ruído de 1 e o corte para o tamanho final de 3 segundos: 47.19% e 22.47% de precisão para o *gzip* e *bzip2*, respetivamente.

Através do cálculo das médias das precisões para cada compressor, com os restantes fatores iguais, podemos verificar a proximidade de precisão referida, tal como é possível observar abaixo, sendo que o *gzip* obteve uma precisão ligeiramente mais alta.

$$gzip : \bar{\mathbf{x}} = \frac{100 + 100 + 96.63 + ... + 94.38 + 76.40}{16} = 67.48\%$$

$$bzip2 : \bar{\mathbf{x}} = \frac{100 + 100 + 94.38 + ... + 94.38 + 77.53}{16} = 63.06\%$$

5 Conclusão

Com este projeto foi possível aprofundar e adquirir conhecimentos acerca da *Normalized Compression Distance*, NCD, e acerca do processo de criação de uma ferramenta de reconhecimento de músicas.

Através dos testes efetuados, conseguimos perceber a influências dos diversos fatores na precisão da identificação das músicas, tais como o tamanho da amostra da música, o tipo de compressor ou o nível de ruído aplicado. Concluindo, podemos afirmar, que o programa conseguiu ter uma precisão consideravelmente alta em grande parte dos casos.