Universidade de Aveiro
Departamento de Eletrónica, Comunicações e Informática

Algorithmic Information Theory (2023/24)

Lab work nº 3

10th June 2024

Rafael Pinto, nº 103379

Ricardo Antunes, nº 98275

Pompeu Costa, nº 103294

Contents

Introduction 2
Implementation 3
Results 4
  Samples without noise 4
  Samples with noise 5
Conclusion 6

Introduction

This report addresses the application of Normalized Compression Distance (NCD) for the identification of music segments. This experiment aims to develop and test a system capable of identifying music names using just short audio samples of that music. The music audio files are transformed into signatures that are more suitable for compression-based comparisons. Then, by computing the NCD between a given sample's signature and the musics' signatures stored in a database, the system identifies the music that yields the smallest NCD value, indicating highest similarity.

We tested the system with various compressors and evaluated its performance using various metrics. Additionally, we tested the robustness of the technique by adding white noise to the audio samples and comparing the results for these samples with the previous ones.

Implementation

For this experiment, it was created seven scripts (can be found in the *scripts* directory):

- *signatures.py* - generates all the signatures for the musics.
- *database.py* - creates the database with the table *Music* that associates a music name to its signature.
- *associate_samples.py* - creates the table *Sample* that associates a manually created sample file to the music where it was taken from.
- *samples.py* – generates and adds samples to the Sample database. For every music extracts its id, slices them into multiple 10 to 20 seconds samples with 5 to 10 seconds between them and associates the created file to the music id.
- *noise.py* – adds white noise to all the samples, saves it in a file and adds that file to the Sample database.
- *labwork3.py* – given an audio file identifies the name of the music in the database that yields the smallest value of the NCD and if the audio file exists in the database, then verifies if the result is correct. A more detailed explanation of this script will be explained further.
- *results.py* – generates results csv files for all samples with all different compressors and calculates various metrics.

Figure 1 shows the diagram of our database, we store the music names and its signatures and when creating samples we associate the sample file name to the music id.

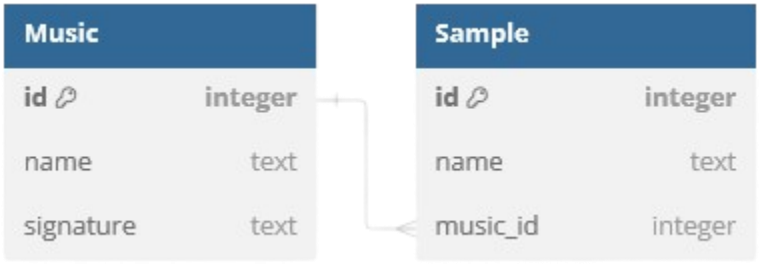

Figure 1 Database Diagram

The *labwork3.py* script is the one that allows identifying a music name given a segment of a certain music. The script accepts as command line arguments the segment file name, the compressor to be used (gzip, lzma or bzip2) and the output file name. It executes the *GetMaxFreqs* program, provided by the professors, with the segment file and stores a temporary file with the signature, then calculates the NCD of the segment's signature with all the musics' signatures in the database. Finally, it computes the minimal value of the NCD and, if the sample file exists in the database, verifies if the result music is correct. The README.md file contains an example demonstrating how to run the script.

Results

The musics used in this experiment were taken from Youtube and the links can be found in *./musics/musics.md* file. The first 50 samples were created manually when adding new music files to the database, the rest were created using the *samples.py* and *noise.py* scripts. It was created 1340 samples, half without noise and half with noise, that can be found in the Onedrive folder linked in *./samples/samples.md* file. Some examples are also in the git repository.

To assess the robustness of the technique we performed tests for samples without noise and with noise. The segments of the musics used are identical, with the only difference being the addition of noise. We did this, in order to the results not being influenced by variations in the music segments themselves.

Samples without noise

Table 1 shows the classification results for samples without noise when using the gzip compressor. With this compressor the accuracy was 85%, precision was 95%, recall was 87% and the F1 score was 89%.

Table 1 gzip classification results for samples without noise

	Correct	**Wrong**
Number of Samples	572	98

Table 2 shows the classification results for samples without noise when using the lzma compressor. With this compressor the accuracy was 68%, precision was 92%, recall was 81% and the F1 score was 82%.

Table 2 lzma classification results for samples without noise

	Correct	**Wrong**
Number of Samples	460	210

Table 3 shows the classification results for samples without noise when using the bzip2 compressor. With this compressor the accuracy, precision, recall and F1 score were all of 99%. As the results show, the bzip2 performed better in identifying similarity between the samples' signatures and the musics' signatures.

Table 3 bzip2 classification results for samples without noise

	Correct	**Wrong**
Number of Samples	668	2

Samples with noise

Table 4 shows the classification results for samples with noise when using gzip compressor. With this compressor the accuracy was 83%, precision was 94%, recall was 86% and F1 score was 88%.

Table 4 gzip classification results for samples with noise

	Correct	**Wrong**
Number of Samples	561	109

Table 5 shows the classification results for samples with noise when using lzma compressor. With this compressor the accuracy was 65%, precision was 92%, recall was 78% and F1 score was 80%.

Table 5 lzma classification results for samples with noise

	Correct	**Wrong**
Number of Samples	438	232

Table 6 shows the classification results for samples with noise when using bzip2 compressor. With this compressor the accuracy was 96%, precision was 97%, recall was 98% and F1 score was 97%. As expected, the results with the samples with noise are worse, but there is not a big difference from the ones with clean samples, proving that this technique is robust enough in handling variations introduced by noise.

Table 6 bzip2 classification results for samples with noise

	Correct	**Wrong**
Number of Samples	648	22

Conclusion

In conclusion, the experiment demonstrates that the chosen technique for identifying music segments based on the NCD is effective. This technique proved to be robust to noise, indicating that it can be used in practical applications in the real world where audio quality may vary. We obtained the best results when using bzip2, proving that this compressor is better in handling repetitive patterns, found in audio frequencies, than gzip and lzma. This experiment contributed to a better understanding of algorithmic information theory and its practical applications, such as in music recognition.