Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática

Algorithmic Information Theory (2021/22)

Lab work nº 3 — Due: 31 Jan 2022

The Normalized Compression Distance (NCD) is an approximation, using compression, of the non-computable Normalized Information Distance (NID), defined as

$$\mathrm{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}},$$

where $K(x)$ is the Kolmogorov complexity of string x and $K(x|y)$ is the Kolmogorov complexity of x when y is given as additional information. To avoid the non-computability of the Kolmogorov complexity, the NCD uses the approximation

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

where $C(x)$ denotes the number of bits required by compressor C to represent x and $C(x,y)$ indicates the number of bits needed to compress x and y together (usually, the two strings are concatenated). Distances close to one indicate dissimilarity, while distances near zero indicate similarity.

The main objective of this work is to develop and test a setup using the NCD for the automatic identification of musics, using small samples for querying the database (for example, 10 seconds). For an introduction to this topic, see the paper by Rudi Cilibrasi *et al.*, available at `http://homepages.cwi.nl/~paulv/papers/music.pdf`,

> Rudi Cilibrasi, Paul Vitányi, and Ronald de Wolf, *Algorithmic Clustering of Music Based on String Compression*, Computer Music Journal, 28:4, pp. 49–67, 2004.

The problem can be stated as follows. Suppose that a certain database contains representations of several (complete) musics, denoted m_i, and that we want to identify a segment of music, represented by string x. The idea is to compute the values of $\mathrm{NCD}(x, m_i), \forall_i$, and assign to x the name of the music in the database that yields the smallest value of the NCD.

For performing the compression, standard compressors, such as `gzip`, `bzip2`, `lzma`, ..., should be used. The report should include results using several compressors. The database should

contain, at least 25 different musics. Testing should be done on small segments of the musics. It is also suggested to add some noise to the segments used for querying, in order to assess the robustness of the technique. For manipulation of audio files, such as format conversion, extraction of segments, addition of noise, etc., see, for example, `http://sox.sourceforge.net/`.

For this approach to be successful, the audio files need to be first converted into a representation (a kind of signature) more adequate to the general purpose compressors that are going to be used. One of the possibilities, is to transform the audio file into the set of its most significant frequencies. Because, generally, the audio frequencies vary along time, this is attained by first cutting the audio into (partially overlapping) segments and then compute the most significant frequencies for each of these segments. An example, named `GetMaxFreqs`, in C++, is available in the website of the course. This example can be used as is or adapted, if required. Of course, independent implementations will be valued.