Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática

Algorithmic Information Theory (2021/22)

Lab work nº 1 — Due: 8 Nov 2021
(to be presented in the class of 11 Nov 2021)

1 Information models for prediction

Usually, the purpose of studying data compression algorithms is twofold. The need for efficient storage and transmission of data is often the main motivation. However, underlying every compression technique, there is a **model** that tries to reproduce as closely as possible the **information source** to be compressed. This model may be interesting on its own, as it can shed light on the **statistical properties (or, more generally, algorithmic properties)** of the source.

One of the most used approaches for representing **data dependencies** relies on the use of Markov models. In lossless data compression, we use a specific type, called discrete time Markov chain or **finite-context model**.

A finite-context model can be used to collect high-order statistical information of an information source, assigning a probability estimate to the symbols of the alphabet, according to a conditioning context computed over a finite and fixed number of past outcomes. Our main goal is to **predict the next outcome** of the source. To do this, we infer a model based on the observed past of the sequence of outcomes.

1.1 Markov models

Let us denote by $x_1^n = x_1 x_2 \ldots x_n,\ x_i \in \Sigma$, the sequence of outputs (symbols from the source alphabet Σ) that the information source has generated until instant n. For example, if $\Sigma = \{0, 1\}$, then we may have $x_1^8 = 01001101$. In this example, we have $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, and so on. A k-order Markov model verifies the relation

$$P(x_n | x_{n-1} \ldots x_{n-k}) = P(x_n | x_{n-1} \ldots x_{n-k} \ldots),$$

where the sub-sequence $c = x_{n-1} \dots x_{n-k}$ is called the state or **context** of the process. A first-order Markov model reduces to

$$P(x_n|x_{n-1}) = P(x_n|x_{n-1}x_{n-2}\dots).$$

Markov models are particularly useful in text compression (but not only!), because the next letter in a word is generally heavily influenced by the preceding letters. In fact, the use of Markov models for written English appeared in the original work of Shannon. In 1951, he estimated the entropy of English (i.e., the average amount of information) to be in between about 0.6 and 1.3 bits per letter.

We can use the frequency of the pairs of letters to estimate the probability that a letter follows any other letter. This reasoning can also be used for constructing higher-order models. However, the size of the model grows exponentially. For example, to build a third-order Markov model, we need to estimate the values of $P(x_n|x_{n-1}x_{n-2}x_{n-3})$. If a table is used[1], it has to accommodate $|\Sigma|^{k+1} = 27^4 = 531\,441$ entries, besides having to rely on enough text to correctly estimate the probabilities.

1.2 The estimation of probabilities

The idea is to collect counts that represent the number of times that each symbol occurs in each context. For example, suppose that, in a certain moment, a binary ($|\Sigma| = 2$) source is modeled by an order-3 finite-context model represented by the table

x_{n-3}	x_{n-2}	x_{n-1}	$N(0\|c)$	$N(1\|c)$
0	0	0	10	25
0	0	1	4	12
0	1	0	15	2
0	1	1	3	4
1	0	0	34	78
1	0	1	21	5
1	1	0	17	9
1	1	1	0	22

where $N(s|c)$ indicates how many times symbol s occurred following context c. Therefore, in this case, we may estimate the probability that a symbol "0" follows the sequence "100" as being $34/(34 + 78) \approx 0.30$.

This way of estimating the probability of an event, e, based only on the relative frequencies

[1]Usually, more sophisticated data structures, such as hash-tables, are used.

of previously occurred events,

$$P(e|c) \approx \frac{N(e|c)}{\sum\limits_{s\in\Sigma} N(s|c)},$$

suffers from the problem of assigning zero probability to events that were not seen during the construction of the model. That would be the case, in this example, if we used the same approach to estimate the probability of having a "0" following a sequence of three or more "1s" (can you see what is the problem?).

This problem is overcome using a "smoothing" parameter for estimating the probabilities, i.e.,

$$P(e|c) \approx \frac{N(e|c) + \alpha}{\sum\limits_{s\in\Sigma} N(s|c) + \alpha|\Sigma|},$$

where α is the smoothing parameter.

1.3 The entropy of the model

Having the model computed, we can estimate the amount of information that a new event, e, needs to be represented. This is given by

$$-\log P(e|c),$$

where c is the context in which e occurs. Note that each row of the table is, *per se*, a statistical model, with **entropy** given by

$$H_c = -\sum_{s\in\Sigma} P(s|c) \log P(s|c).$$

The overall entropy of the model is the weighted average of the entropies of each state (or contexts), where the weights are the probabilities of those states, i.e.,

$$H = \sum_{c} P(c) H_c,$$

where $P(c)$ denotes the estimated probability of state (context) c.

2 Work to be done

1. Develop a program, named `fcm`, with the aim of collecting statistical information about texts, using finite-context models. The order of the model, k, as well as the smoothing parameter, α, should be parameters passed to the program. This program should provide the entropy of the text, as estimated by the model.

2. Using `fcm` as a starting point, develop a program for automatic text generation (named `generator`) that, based on a finite-context model learned beforehand (using a text as example), is able to generate text that follows that statistical model.

3. Elaborate a report, where you describe all the steps and decisions taken during the development of the work. Include relevant and illustrative results that were obtained. In particular, a discussion regarding the effects of the variation of the parameters, and also how different types of texts compare in terms of entropy, should be included. Besides the texts that you choose, it is mandatory to include entropy measures for the text `example.txt`, available in moodle.

3 How to deliver the work

The work should be developed and made available to the teachers using a repository following the structure suggested in Fig. 1 (if needed, further instructions will be provided during the classes). The developed software should be compatible with the Linux operating system. It should include a README file, explaining, with sufficient detail, how to build the program and all the parameters that it accepts. It should also include, at least, how to run a concrete example.

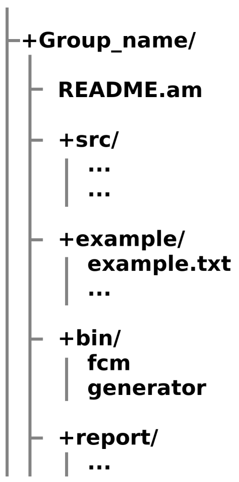

Figure 1: Suggested repository structure.