UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

Algorithmic Information Theory (2024/2025)

Lab Work #1

Deliver: 10 Mar 2025 (presentation in class: 14 Mar 2025)

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 (or the next symbol of a sequence). 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 $x_1^n = x_1 x_2 \dots x_n$, $x_i \in \Sigma$, 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, $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, and so on. A k-order Markov model verifies the relation

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

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 symbol in a word is generally heavily influenced by the preceding symbols. 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 between about 0.6 and 1.3 bits per symbol.

We can use the frequency of the pairs of symbols to estimate the probability that a symbol follows any other symbol. 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, it has to accommodate $|\Sigma|^{k+1} = 27^4 = 531,441$ entries, besides having to rely on enough text to correctly estimate the probabilities. Therefore, for higher k, a memory efficient structure is recommended, such as a **hash table**.

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

$$\frac{34}{34+78} \approx 0.30.$$

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

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

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.,

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

where α (**alpha**) is the smoothing parameter.

1.3 The Average Information Content of a Sequence

The theoretical Average Information Content of a sequence provided by a finite-context model, after having processed n symbols, is given by

$$H_n = -\frac{1}{n} \sum_{i=1}^{n} \log_2 P(x_i|c) \text{ bps}, \tag{2.1}$$

where "bps" stands for "bits per symbol".

It is important to note that the lower the average information content, the better the model fits the sequence. This indicates that the model provides a more accurate description of the sequence, leading to a more effective data without any loss of information.

2 Work to be Done

1. Develop a program in C, C++, or Rust (recommended languages), named **fcm**, to measure the information content of 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 Average Information Content of the text, as estimated by the model.

2. Develop an automatic text generator, named **generator**, that generates text following a learned model. The idea is to collect the counts according to a sequence. Then, freeze the model counts and generate text according to a given context start (prior). The size of the sequence to generate, prior, α, and context must be parameters to the program.

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 average information content, should be included. Besides the texts that you choose, it is mandatory to include average information measures for the text examples (sequence1.txt, sequence2.txt, ... , sequence5.txt), available in moodle (Lab Work 1 in "Sequences for Analysis").

3 How to Deliver the Work

Send an email to both teachers (an@ua.pt and pratas@ua.pt), with all authors in CC, containing the compressed repository or the project link. If the repository is private on GitHub, include an invitation.

The repository must contain:

- Source code for the `fcm` and `generator` programs.
- Necessary files for testing, including example text files.
- A `README.md` file with the following information:
  - **Installation Instructions**: Describe how to compile the program on a Linux machine, including any dependencies.
  - **Running the Programs**: Provide example commands to run the `fcm` and `generator` programs, such as:
    - For `fcm`: `./fcm input_text.txt -k 3 -a 0.01`
    - For `generator`: `./generator -k 3 -a 0.1 -p abc -s 500`
  - **Compilation Instructions**: Include the exact commands to compile the programs, such as: `make` or `gcc`.
  - **Dependencies**: List any required libraries or tools and how should they be installed (the exact commands).
- Provide the Report in **PDF** format only.