Computing entropy of data

Information Theory Lab 2

Objective

Understand the concepts of entropy and discrete memoryless source. Model a data file as a sequence of independent 1-byte random variables and compute its entropy.

Theoretical notions

The entropy of a discrete memoryless source is defined as:

$$H(S) = \sum_i p(s_i) \cdot \log_2(p(s_i))$$

See the lecture notes for more details.

Practical issues

A file is a sequence of 1-byte values (i.e. between 0 and 255).

If we ignore all interdependencies in the data, we can consider the bytes as **independently drawn from a discrete memoryless source** with 256 different possible messages.

- Note: this is a (simplifying) model, in reality the data values are not independent, but depending strongly on the surrounding values.

The distribution of the source can be estimated by counting the number of apparitions of every byte value in the file, and dividing to the total number of bytes. For example, if the byte value 55 appears 1000 times in a 9000-byte file, its probability is 1/9.

Exercises

1. Write a C program to compute the entropy of a file.

- The program shall receive the name of the file as a command-line argument: `entropy.exe myfile.txt`
- The program should follow the following steps:
  - Open the file for reading (in binary format)
  - Count the number of apparitions of every byte value:
    - hold an array of 256 counters, one for every possibly byte value
    - repeatedly read one byte from the file
    - increment the counter corresponding to the byte read
    - also store and increment a counter for the total number of bytes
  - Compute the probability of every byte value: divide each byte counter to the global counter
  - Compute the entropy, based on the probabilities
  - Show the result

1. Use the program to estimate the entropy of the following data files:
   - a Romanian language text (`textRO.txt`);
   - an English language text (`textRO.txt`);
   - an executable file (*.exe)
   - a JPG image file
2. Considering the computed entropy for `textRO.txt`, estimate how many bits would be sufficient to encode the file, and compare with the actual file size. Archive the file in the *.zip* format and compare the resulting size with your estimated size.
3. Modify the program to list the probabilities of the letters `a` to `z`. Compare the probabilities for Romanian and English. Are they different? For what letters?

Implementation hints

- The counters can be implemented as a vector of integers, with size 256, such that `counter[i]` is the counter for the byte value `i`. When the value `i` is read from the file, incrementing the corresponding counter is done easily with `counter[i]++`.
- You will need another vector for probabilities (real values).
- The following C functions may be used for file-based operations. Look up their documentation on the Internet (e.g. *cplusplus.com*, or Google search).
- `fopen(...)`, to open a file for reading;
- `fread(...)`, to read byte data from the file;

- `fclose()`, to close the file when finished.
- There is no function in C for computing $\log_2(x)$. You can compute this using the logarithm conversion formula:

$$\log_2(x) = \frac{\log_b(x)}{\log_b(2)},$$

using one of the existing logarithms in C: `log()` (natural logarithm) or `log10()` (logartithm base 10).

Final questions

1. Suppose we consider pairs of two consecutive bytes as the messages of the source. What are the drawbacks of this implementation? How will the computed entropy be in this case?
2. Repeatedly reading 1 byte from a file on the disk is inefficient. Can we improve the program by reading data in larger chunks? Think / describe what needs to be changed.