Playing with data

Tom Rochette <tom.rochette@coreteks.org>

May 6, 2018 — 1975149b

0.1 Learned in this study

0.2 Things to explore

- Symbols
- Data stencils

1 Overview

1.1 Symbols

Information does not exist without structure. Information cannot exist without consistency. If I write a word and the strokes of the letters vary too much, making it impossible to relate two instances of the same word, then it is not possible to use experience to relate to priors.

Without defining the atomic units we are working with, one cannot extract value from a data stream.

One can certainly apply dictionaries of symbols, mapping an input (sequence of bits) with a comprehensible output (a character for instance, or a pixel description).

The question at this point becomes, how can we obtain/generate as many dictionaries as possible?

One way to do so is through experience acquisition, that is, by trying things out and recording the result. By doing so, an agent may slowly build a map of inputs/outputs which he may be able to reuse in the future.

We're also curious about how to determine quickly which of all of our dictionaries is the most appropriate for a data stream (as we certainly would like to avoid processing the data stream using dictionaries while generating output garbage). In this case there might be a few ways to prevent or reduce the amount of "garbage" data generated.

One solution is to produce the stream of output for each dictionary and evaluate each stream after a time t. The purpose of the evaluation is to determine if the generated output stream is noise or it is producing something that can be part of a whole. For instance, if one dictionary is producing formulas but the produced formula so far is already malformed, we can say it is highly unlikely that this is the appropriate dictionary. On the other hand, if the dictionary is building an encrypted string, then it is close to impossible to evaluate if said string is part of a whole or not.

This idea implies that we already have enough knowledge about the syntax (structure) and semantics of the dictionary's language that we can assert whether something is valid or not. It is even possible to do that?

1.1.1 Symbols pipelines

The idea of symbols pipeline is that a data stream can be converted into various outputs. For instance, an image can be converted into various image formats, it can be converted into an ASCII art text, a textual description, a single number, etc.

As such, the purpose of the symbol pipeline is to convert a data stream from one representation to another.

1.2 Data stencils

One interesting aspect of data is how it is interpreted. The same bits of data can be understood differently depending on how the reader processes them (which translation space is used).

For example, let's imagine two programs that read the same source of data. The first program reads unsigned 32 bits numbers and sums them. The result is the sum of all the data it has seen.

A second program reads in the same fashion, 32 bits words, and adds them to one another. The exception is that in this program, numbers are considered signed instead of unsigned. If certain numbers are negative, then the output of both programs will be different. However, if all numbers were positive, then both programs will behave the same, up to the point where the program using signed numbers will overflow and thus produce a negative value.

A data stencil is a definition of how a program may read a blob of data. The data file may have a header with information and a body of content, it may contain fixed or variable length symbols, etc.

Our goal here is to be able to find that stencil when looking at data. For instance, if I give you a text file without extension (only binary), then a program should be able to find the appropriate stencil for it (utf8) and apply it, returning us the content of interest.

If I give it an image or a video, the same should apply.

A more appropriate name for data stencils would be "a common format". As long as the representation space is shared by both parties (sender and receiver, writer and reader, encrypter and decrypter), we are able to understand each other.