Does the Shannon Bound Really Apply to Data Structures?

Antti Valmari
Tampere University of Technology
FINLAND

1 Introduction
2 Classic Information Theory
3 A Weird Data Structure
4 Formalization of Data Structures
5 Theorems in the Paper
6 Discussion
Introduction

• Consider representing a value drawn from a set of possible values
• Information theory tells how many bits are needed on the average
  – different values may use different numbers of bits
  – a white flag contains little information — except at war!
• Classic information theory uses contiguous bit strings
• Data structures are not like that
• Of course, extending classic information theory to data structures is trivial, isn’t it?
  – e.g., list all bits in a sequence — but xor-pointers?
  – e.g., list visited bits in a sequence — but does that suffice?
    “the value” is the whole phonebook, not an individual name–number pair
  – well, can’t do it now but everyone knows that it is trivial
• My claim is that it is not trivial but not very hard either
  – it is at the right level for a symposium paper and talk
Introduction

• Is it really obvious that it should be trivial?
  – McMillan’s 1956 extension of the application area of Shannon’s bound:
    “... for any complex $x$ such that $|xa| < 1$ the infinite series
    $1 + N(1)x + N(2)x^2 + \cdots$ converges ...”
  – (the results in my paper are more trivial than that)

• The results in my paper:
  – an example demonstrating that the classic proof fails for data structures
    $\Rightarrow$ solid (even if simple) theory is needed instead of just handwaving
  – a suitable rigorous (and simple) definition of data structures
  – a theorem that extends classic theory to all non-weird data structures
  – the observation that given the above, the bound for weird d.s. is trivial ...
  – ... but non-optimality of redundant representations is not:
    *unlike in classic theory, it need not hold if probabilities may be 0
  – a non-optimality theorem for redundant representations if all $p_i > 0$

• To discuss these, first we need to look at classic information theory
Classic Information Theory

• If all values are equally likely
  – 8 bits suffice to represent $2^8 = 256$ different values
  – to represent one of $N$ values, $\lceil \lg N \rceil$ bits are needed

• If value $i$ has probability $p_i$ and representation (= codeword) $r_i$
  – short codewords for common and long codewords for rare values
  – e.g., Tallinn = 00, Tartu = 01, Pärnu = 1
  – average memory usage $= \sum_{i \in I} p_i |r_i|$

• Unique decodability
  – a word in Finnish

• Self-delimiting bit strings (= prefix codes)
  – no codeword is a prefix of another
  ⇒ codeword can be uniquely decoded immediately after reading its last bit
Classic Information Theory

• **Shannon:** with self-delimiting bit strings at least

\[- \sum_{i \in I} p_i \log_2 p_i\]

bits are needed on the average

• The only thing the proof assumes of the encoding is **Kraft’s inequality**

\[\sum_{i \in I} 2^{-|r_i|} \leq 1\]

• **McMillan:** Kraft’s inequality holds for *all* uniquely decodable bit string sets

\[\Rightarrow\] also the \[- \sum_{i \in I} p_i \log_2 p_i\] bound holds

– McMillan’s proof is unexpectedly difficult!

• Unlike data structures

– never two codewords for the same value

  (would strictly increase memory consumption, even if \( p_i = 0 \))

– all codewords are contiguous bit strings with clear start and end
A Weird Data Structure

- A concurrent system
  - synchronous communication (like Ada, Occam, CSP)
  - no sub-sub tasks
- Master's view to activity bits
  
  \[
  \begin{align*}
  00 & \quad = \text{can switch off} \\
  01, 10, 11 & \quad = \text{must still wait}
  \end{align*}
  \]

  \[\Rightarrow\] one bit of information

- A modification saving memory
  
  - when a server is serving a sub-task, the other is serving an original task
  \[\Rightarrow\] the activity bit of the other is certainly 1

  \[\Rightarrow\] the master gets the right result independently of the bit of the first server

  \[\Rightarrow\] the first server can freely use the bit for something else

- When serving a sub-task, a server uses its activity bit for something else than denoting activity, and this does not fool the master
A Weird Data Structure

• Let – denote that the activity bit is used for something else
  – master does not know that, and may accidentally read it getting 0 or 1
  – the other use determines the value absolutely freely as it needs
  ⇒ the bit must not be counted into the memory consumption of activity info

⇒ Memory consumption
  – 00, 01, 10, 11: two bits
  – 1–, –1: one bit

• Put these numbers to Kraft’s inequality

\[
\sum_{i \in I} 2^{-|r_i|} = 2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} + 2^{-1} + 2^{-1} = 2 > 1
\]

• Leave 01, 10, and 11 out, since they are covered by 1– and –1

\[
\sum_{i \in I} 2^{-|r_i|} = 2^{-2} + 2^{-1} + 2^{-1} = \frac{1}{4} > 1
\]

⇒ Kraft’s inequality is violated

• The essence here is that 1– and –1 are two distinct representations of size 1
Formalization of Data Structures

• What is a data structure?
  – uses memory to represent one value from a set of possible values

• What is a representation?
  – uses some bits of memory and does not use the remaining bits
  – assigns 0 or 1 to each used bit
  ⇒ can be formalized as \( r = (B_0, B_1) \) such that \( B_0 \cap B_1 = \emptyset \)
  – the elements of \( B_0 \) and \( B_1 \) are, e.g., memory addresses

• The same value may have many representations
  – this happens all the time with data structures

• Representations of different values must be distinguishable from each other
  – if \( (B_0, B_1) \) and \( (B'_0, B'_1) \) represent different values, they must have a conflicting bit
  ⇒ we require \( B_0 \cap B'_1 \neq \emptyset \) or \( B_1 \cap B'_0 \neq \emptyset \)
Theorems in the Paper

- A representation scheme is fully conflicting iff each pair of distinct representations has a conflicting bit
- **Theorem** If it is fully conflicting, then Kraft’s inequality holds.
  \[ \Rightarrow \text{violating Kraft’s inequality requires two indistinguishable representations for the same value (like } 1^- \text{ and } -1 \]  
  - proof of theorem: \( \frac{1}{2} \) pages on the paper, induction on \( |R| \), from \( R \) to \( R_0, R_1, R_\perp \) to \( R'_0 \cup R_\perp, R'_1 \cup R_\perp \)

- All non-weird data structures are now covered
- Trivial observation: leaving out redundant representations does not increase average memory consumption, if the one that is kept is minimal
  - yields a fully conflicting representation scheme
  \[ \Rightarrow \text{Shannon’s bound holds} \]

- Also the weird data structures are now covered “bound-wise”
- However, another important classic result deserves new analysis
Theorems in the Paper

• In classic theory, redundant representations always imply non-optimality
  – deepens the significance of the bound

⇒ Of course I wanted to prove the same for data structures

• Observation: it does not hold fully generally

• **Theorem** If at least one object has two representations, and if the probability of every representation is \( > 0 \), then the scheme is not memory-optimal.
  – proof of theorem: 1 page on the paper + \( \frac{1}{2} \) pages to support intuition,
    construct two fully conflicting schemes such that \( \mathcal{U}(R) \) and \( R \) use the same amount of memory, and
    \[
    1 \geq \sum_{r \in \mathcal{E}(R)} 2^{-|r|} > \sum_{r \in \mathcal{U}(R)} 2^{-|r|}
    \]

• For weird data structures, used redundancy ⇒ non-optimality

• Classic representations qualify as data structures

• An optimal self-delimiting representation exists in classic theory

⇒ The bound is strict also with data structures
Discussion

• A hole in the proof of a very fundamental result was revealed and fixed
  – said like that, it sounds incredible — is the paper really okay?

• Anticipated objection: It must be trivial, the paper just makes it complicated
  – I still have not seen a trivial argument

• Let’s try to make it trivial: If the same value has many representations, only a
  shortest should be taken, and then Kraft’s inequality holds
  – some justification is needed that then Kraft’s inequality holds
  – to do that, we need to say what we mean by a data structure
    = Definition 5 and Theorem 6 of the paper
    ⇒ more or less the same reasoning as in the paper

• Furthermore, “redundancy ⇒ non-optimality” was brought to data structures
  – had to add the (rather weak) assumption “used redundancy”

• The paper sorts out an unimportant but real pathological case
Discussion

• Is this a major result? No!
  – Shannon’s bound for data structures was not changed, only its proof
    ⇒ does not affect anyone
  – the situation where Kraft’s inequality was violated is unusual
  – the redundancy theorem remains impossible to cheat
  – although the proofs are not trivial, they are not horribly difficult either
    (much of the paper is not proofs but telling why they are needed)

• So, is this a zero result? Not in my opinion.
  – faith in the bound and “used redundancy ⇒ non-optimality”
    for data structures need not any more be based on handwaving

• Perhaps this is a result that inspires a lively after-talk discussion?
  – unless I have overrun my time slot . . .

Thank you for attention!