The Value of Interaction in Data Intelligence

MIN CHEN, University of Oxford, UK

Fig. 1. Data processing inequality (DPI) is a mathematical theorem that confirms and underlines the challenge of information loss in almost all fully automatic data intelligence workflows except some simple ones with very small data spaces. Human computer interaction (HCI) can provide cost-beneficial means to alleviate the problems due to DPI.

In human computer interaction (HCI), it is common to evaluate the value of HCI designs, techniques, devices, and systems in terms of their benefit to users. It is less common to discuss the benefit of HCI to computers. Every HCI task allows a computer to receive some data from the user. In many situations, the data received by the computer embodies human knowledge and intelligence in handling complex problems, and/or some critical information without which the computer cannot proceed. In this paper, we present an information-theoretic framework for quantifying the knowledge received by the computer from its users via HCI. We apply information-theoretic measures to some common HCI tasks as well as HCI tasks in complex data intelligence processes. We formalize the methods for estimating such quantities analytically and measuring them empirically. Using theoretical reasoning, we can confirm the significant but often undervalued role of HCI in data intelligence workflows.

CCS Concepts:
- Human-centered computing → HCI theory, concepts and models; Visualization theory, concepts and paradigms;
- Mathematics of computing → Information theory; Information systems → Data analytics.

Additional Key Words and Phrases: Human-computer interaction, information theory, cost-benefit, interaction, knowledge, visualization, data intelligence.

Author’s address: Min Chen, min.chen@oerc.ox.ac.uk, University of Oxford, UK.
1 INTRODUCTION

Data intelligence is an encompassing term for processes that transform data to decisions or knowledge, such as statistical inference, algorithmic analysis, data visualization, machine learning, business intelligence, numerical modelling, computational simulation, prediction, and decision making. While many of these processes are propelled by the desire for automation, human-computer interaction (HCI) has been and is still playing valuable roles in almost all nontrivial data intelligence workflows. However, the benefits of HCI in a data intelligence workflow are often much more difficult to measure and quantify than its costs and disadvantages. This inevitably leads to a more fervent drive for replacing humans with machines in data intelligence.

In information theory, the Data Processing Inequality (DPI) is a proven theorem. It states that fully automated processing of data can only lose but not increase information. As Cover and Thomas explained [12], “No clever manipulation of data can improve the inferences that can be made from the data.” In most data intelligence workflows, since the original data space contains much more variations (in terms of entropy) than the decision space, the loss of information is not only inevitable but also can be very significant [8].

In the context of data visualization, Chen and Jänicke first pointed out that HCI alleviates the undesirable bottleneck of DPI because the mathematical conditions for proving DPI are no longer satisfied with the presence of HCI. As illustrated in Fig. 1(a), the proof of DPI assumes that (i) each process in a workflow must receive data only from its proceeding process, and (ii) the output of a process must depend only on the output of its proceeding process. As illustrated in Fig. 1(b), any human inputs based on human knowledge (e.g., the variation of context and task) violate the first condition. Meanwhile any human inputs based on observing the previous processes in the workflow (e.g., the details being filtered out or aggregated) violate both conditions. Therefore, if we can quantitatively estimate or measure the amount of information passing from humans to the otherwise automated processes, we can better appreciate the value of interaction in data intelligence.

In this paper, we present an information-theoretic framework for measuring the knowledge received by a computational process from human users via HCI. It includes several fundamental measures that can be applied to a wide range of HCI modalities as well as the recently discovered cost-benefit measure for analyzing data intelligence workflows [4, 8, 10]. We describe the general method for estimating the amount of human knowledge delivered using HCI. We outline the general design for an empirical study to detect and measure human knowledge used in data intelligence. With these theoretical contributions, we can explore the value of HCI from the perspective of assisting computers, which differs from the commonly-adopted focuses on assisting human users.

2 RELATED WORK

In the field of HCI, the term of “value” has several commonly-used referents, including (a) worth in usefulness, utility, benefit, merit, or importance, (b) monetary, material, developmental, or operational cost, (c) a quantity that can be measured, estimated, calculated, or computed, and (d) a principle or standard in the context of moral or ethics. In this paper, we examine the value of HCI processes primarily in terms of (a) and (c) with some supplemental discussions on (b). Readers who are interested in (d) may consult other works in the literature (e.g., [16, 29, 34, 38]).

Most research effort in HCI has been devoted to bring about the usefulness and benefits to humans. The goals of HCI and the criteria for good HCI are typically expressed as “support people so that they can carry out their activities..."
The Value of Interaction in Data Intelligence

productive and safely” [26]; “effective to use, efficient to use, safe to use, having good quality, easy to learn, easy to remember how to use” [27]; “time to learn, speed of performance, rate of errors by users, retention over time, subjective satisfaction” [37]; and “useful, usable, used” [14]. On the contrary, it is less common to discuss the usefulness and benefits of HCI to computers. While there is little doubt that the ultimate goal is for computers to assist humans, it will be helpful to understand and measure how much a computer needs to be assisted by its users before becoming useful, usable, and used. It is hence desirable to complement the existing discourses on value-centered designs (e.g., [2, 11, 15, 22, 29]) by examining the value of HCI from the perspective of computers.

It is also feasible to develop quantitative methods for measuring and estimating how much a computer needs to know since we can investigate the inner “mind” of a computer program more easily than that of human users. The field of HCl has benefited from a wide-range of quantitative and qualitative research methods [3, 13, 17, 19, 21, 23–25, 28], including quantitative methods such as formal methods, statistical analysis, cognitive modelling, and so on. This work explores the application of information theory [12, 32] in HCI.

Claude Shannon’s landmark article in 1948 [32] signifies the birth of information theory. It has been underpinning the fields of data communication, compression, and encryption since. Its applications include physics, biology, neurology, psychology, and computer science (e.g., visualization, computer graphics, computer vision, data mining, and machine learning). The cost-benefit measure used in this work was first proposed in 2016 [8] in the context of visualization and visual analytics. An improvement was proposed recently to make the interpretation of the numerical quantification more consistent with practical observations [4, 10]. It has been used to prove mathematically the correctness of a major wisdom in HCI, “Overview first, zoom, details-on-demand” [6, 9, 35], and to analyze the cost-benefit of different virtual reality applications [7]. Two pieces of previous work have showed that the measure can be estimated in practical applications [39] and be measured using empirical studies [20], while other pieces have demonstrated its uses in qualitative analysis of data intelligence workflows [5] in general, and in particular, in improving the designs of dashboards [1], visualization for ensemble machine learning [41], and sensitive analysis of epidemiological models [30]. While information theory has been applied successfully to visualization (i.e., interaction from computers to humans), this work focuses on the other direction of HCI (i.e., from humans to computers).

3 FUNDAMENTAL MEASURES

From every human action through a user interface or an HCI device, a computer receives some data, which typically encodes information that a running process on the computer wishes to know, and cannot proceed without. Through such interactions, humans transfer their knowledge to computers. In some cases, the computers learn and retain part of such knowledge (e.g., preference setting and annotation for machine learning). In many other cases, the computers asked blithely for the same or similar information again and again.

In HCI, we all appreciate that measuring the usefulness or benefits of HCI to humans is not a trivial undertaking. In comparison, the amount of knowledge received by a computer from human users via HCl can be measured relatively easily. Under an information-theoretic framework, we can first define several fundamental measures about the information that a computer receives from an input action. We can then use these measures to compare different types of interaction mechanisms, e.g., in terms of the capacity and efficiency for a computer to receive knowledge from users.

3.1 Alphabet and Letter

When a running process on a computer pauses to expect an input from the user, or a thread of the process continuously samples the states of an input device, all possible input values that can be expected or sampled are valid values of
a univariate or multivariate variable. In information theory, this mechanism can be considered in abstraction as a communication \textit{channel} from a user to a computational process. This variable is referred to as an \textit{alphabet}, \( Z \), and these possible values are its \textit{letters}, \( \{z_1, z_2, \ldots, z_n\} \).

In a given context (e.g., all uses of an HCI facility), each letter \( z_i \in Z \) is associated with a probability of occurrence, \( p(z_i) \). Before the process receives an input, the process is unsure about which letter will arrive from the channel. The amount of uncertainty is typically measured with the Shannon entropy [32] of the alphabet:

\[
H(Z) = -\sum_{i=1}^{n} p(z_i) \log_2 p(z_i) \quad \text{[unit: bit]} \tag{1}
\]

We can consider alphabets broadly from two different perspectives, the \textit{input device} and the \textit{input action}, which are detailed in the following two subsections.

### 3.2 Input Device Alphabet

An \textit{input device alphabet} enumerates all possible states of a physical input device, which can be captured by a computational process or thread through sampling. Such devices include keyboard, mouse, touch screen, joystick, game controller, VR glove, camera (e.g., for gestures), microphones (e.g., for voices), and many more. Most of these devices feature multivariate states, each of which is a letter in the input device alphabet corresponding to the device.

For example, the instantaneous state of a simple 2D pointing device may record four values: its current location in \( x \)-\( y \) relative to a reference point, the activation status of its left and right buttons. The instantaneous state of a conventional keyboard may consist of 80-120 variables, one for each of its keys, assuming that simultaneous key activations are all recorded before being sequentialized and mapped to one or more key inputs in an input action alphabet (to be discussed in the next subsection).

The design of an input device may involve many human and hardware factors. Among them, the frequency or probability, in which a variable (e.g., a key, a button, a sensor, etc.) changes its status, is a major design consideration. Hence, this particular design consideration is mathematically underpinned by the measurement of Shannon entropy. The common wisdom is to assign a lower operational cost (e.g., speed and convenience) to a variable that is more frequently changed (e.g., a more frequently-used button). This is conceptually similar to entropic coding schemes such as Huffman encoding in data communication [18].

However, the sampling mechanism for an input device usually assumes an equal probability for all of its variables. From the perspective of the device, any variables may change at any moment, and all letters (i.e., states) in the alphabet have the same probability. This represents the maximal uncertainty about what is the next state of the input device, as well as the maximal amount of information that the device can deliver. For an input device alphabet \( Z \) with \( n \) letters and each letter has a probability of \( 1/n \), this maximal quantity is the maximum of the Shannon entropy in Eq. (1), i.e.,

\[
H_{\text{max}} = \log_2 n = \log_2 \|Z\|. \quad \text{We call this quantity the \textit{instantaneous device capacity} and we denote it as} \ C_{\text{dev}}.
\]

For example, for a simple 2D mouse with 2 on-off buttons, operating in conjunction with a display at a 1920×1080 resolution, its instantaneous device capacity is:

\[
C_{\text{dev}} = H_{\text{max}} = \log_2 1920 + \log_2 1080 + \log_2 2 + \log_2 2 \\
\approx 10.907 + 10.077 + 1 + 1 = 22.984 \text{ bits}
\]

While the notion of instantaneous device capacity may be useful for characterizing input devices with which a sampling process has to be triggered by a user’s action, it is not suited for input devices with a continuing sampling
process (e.g., a video camera for gesture recognition). Hence a more general and useful quantity for characterizing all input devices is to define the maximal device capacity over a unit of time. We use "unit: second" for the unit of time in the following discussions. Let \( r \) be the sampling rate, that is, maximal number of samples that a process can receive from an input device within a second. Assuming that the instantaneous device capacity of the device is invariant for each sample, the bandwidth (cf. bandwidth in data communication) of the device is defined as:

\[
W_{\text{dev}} = r \times \text{instantaneous device capacity} \quad \text{[unit: bit/s].}
\]

Note: while the instantaneous device capacity is measured in bit, the bandwidth, \( W_{\text{dev}} \), is measured in bit per second.

For example, if the sampling rate of the aforementioned mouse is 100 Hz, then its bandwidth is \( W_{\text{dev}} \approx 2,298.4 \) bits/s. Similarly, consider a data glove with 7 sensors with a sampling rate of 200Hz. If its five sensors for finger flexure have 180 valid values each and the pitch and roll sensors have 360 valid values each, its bandwidth is:

\[
W_{\text{dev}} = 200(5 \log_2 180 + 2 \log_2 360) \approx 10,888.6 \text{ bits/s.}
\]

### 3.3 Input Action Alphabet

An input action alphabet enumerates all possible actions that a user can perform for a specific HCI task in order to yield an input meaningful to the computer. Here the phrase "a specific HCI task" stipulates the condition in which the user is aware of what the computer wants to know, e.g., through a textual instruction or a visual cue on the screen or through context awareness based on previous experience or acquired knowledge. The phrase "meaningful to the computer" stipulates the condition in which an action that the computer is not programmed to handle for the specific HCI task should not be included in the input action alphabet.

Consider a simple HCI task of making a selection out of \( k \) radio buttons. (Multiple-choice buttons can also be included in this consideration.) Assuming that selecting nothing is not meaningful to the computer, the corresponding input action alphabet is: \( A_{\text{radio}} = \{\text{option}_1, \text{option}_2, \ldots, \text{option}_k\} \). When each option is associated with a binary bit in an implementation, the letters in the alphabet can be encoded as a set of \( k \)-bit binary codewords: \{00, 001, 010, \ldots, 1000\}. If all \( k \) options are equally probable, the entropy of the alphabet is \( \log_2 k \). A selection action by the user thus allows the computer to remove \( \log_2 k \) bits of uncertainty, or in other words, to gain \( \log_2 k \) bits of knowledge from the user for this specific HCI task. We call this quantity the action capacity of the HCI task, which is denoted as \( C_{\text{act}} \).

In practice, radio buttons featured in many HCI tasks do not have the same probability of being selected. For example, as shown in Fig. 2(a), after selecting a channel from a list of current shows, the TV displays an input action alphabet \( A \).
with three options, \(a_1\): “More Event Info”, \(a_2\): “Select Channel”, and \(a_3\): “View HD Alternatives”. The probability of \(a_1\) depends on several statistical factors, e.g., how informative is the title in the list, how many users prefer to explore a less-known program via investigational viewing verse how many prefer reading detailed information, and so on. The probability of \(a_3\) depends on how often a user selects a non-HD channel from a TV listing with an intention to view the corresponding HD channel. Different probability distributions for \(A\) will lead to different amount of knowledge \(\mathcal{H}(A) = C_{\text{act}}\). As exemplified by the instance below, the more skewed a distribution is, the less the knowledge is worth or the less action capacity that the HCI task has:

\[
p(a_1) = p(a_2) = p(a_3) = 1/3 \implies C_{F_a} \approx 1.58
\]

\[
p(a_1) = 0.2, p(a_2) = 0.7, p(a_3) = 0.1 \implies C_{F_b} \approx 1.16
\]

\[
p(a_1) = 0.09, p(a_2) = 0.9, p(a_3) = 0.01 \implies C_{F_c} \approx 0.52
\]

When the probability of a letter in an alphabet becomes 1, the alphabet no longer has any uncertainty. As shown in Fig. 2(b), if a TV offers only one optional answer, the device capacity of the corresponding alphabet, \(C_{\text{act}}\), is of 0 bits. We will revisit this example in Section 5.

Similarly, we can apply entropic analysis to \textit{check boxes}. Consider an input action alphabet \(A_{\text{checkbox}}\) that consists of \(k\) check boxes. There are \(m = 2^k\) possible combinations: \(A_{\text{checkbox}} = \{\text{combination1}, \text{combination2}, \ldots, \text{combination}_m\}\). The alphabet can be encoded using a \(k\)-bit code, \(b_1b_2b_3 \cdots b_k\), where each bit, \(b_j\), \(1 \leq j \leq k\), indicates whether the corresponding checkbox is on or off. If all combinations have the equal probability, the amount of knowledge that computer can gain from the user is \(C_{\text{act}} = k\) bits, which is also the maximum entropy of the alphabet. Note that \(k > \log_2 k\) when \(k > 1\), indicating that \(k\) checkboxes have more action capacity than \(k\) radio buttons except that they have the same action capacity when \(k = 1\) (assuming that both are allowed to have on-off states).

We now examine a more complicated type of input actions. Consider an HCI task for drawing a \textit{freehand path} using a 2D pointing device. Assume the following implementation constraints: (i) the computer can support a maximum \(m\) sampling points for each path; (ii) the drawing canvas is a rectangular area \([x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}]\); (iii) the points along the path are sampled at a regular time interval \(\Delta t\), though the computer does not store the time of each sample; and (iv) the sampling commences with the first button-down event and terminates with the subsequent button-up event.

Let \(A_{\text{freehand}}\) be all possible paths that a user may draw using the 2D pointing device, and \(A_{\text{freehand}}^{(k)}\) be a subset of \(A_{\text{freehand}}\), consisting of all paths with \(k\) points \((k \in [1, m])\). The sub-alphabet \(A_{\text{freehand}}^{(k)}\) thus enumerates all possible paths in the form of \((x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\) where each point \((x_i, y_i)\) is within the rectangular area \([x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}]\). If it is possible to select any pixel in the rectangular area for every point input, the total number of possible paths, is \((x_{\text{max}} - x_{\text{min}} + 1) \times (y_{\text{max}} - y_{\text{min}} + 1))^k\), which is also the size of the sub-alphabet \(A_{\text{freehand}}^{(k)}\).

For example, given a 512 \times 512 rectangular area, the grand total number of possible paths is:

\[
\|A_{\text{freehand}}^{(k)}\| = \sum_{k=1}^{m} \|A_{\text{freehand}}^{(k)}\| = \sum_{k=1}^{m} 2^{18k} \geq 2^{18m}
\]

If all paths have an equal probability, the maximal amount of knowledge that the computer can gain from a user’s freehand drawing action is thus \(C_{\text{act}} = 18\) bits when \(m = 1\), or slightly more than \(C_{\text{act}} = 18m\) bits when \(m > 1\). For an alphabet of possible paths with up to \(m = 20\) points, the maximal amount of knowledge, \(C_{\text{act}}\), is slightly more
The Value of Interaction in Data Intelligence

than 360 bits. This is similar to the amount of knowledge that a computer would gain from an HCI action involving

2^{360} = 2.3 \times 10^{108} radio buttons or 360 checkboxes.

Many data glove devices come with a built-in gesture recognition facility. The gestures that can be recognized by
such a device are letters of an input action alphabet $A_{\text{gestures}}$. For example, an alphabet may consist of 16 elementary
gestures (1 fist, 1 flat hand, and 14 different combinations of figure pointing). The maximum entropy of this alphabet,
i.e., the maximal amount of knowledge $C_{\text{act}}$ that can be gained, is $H_{\text{max}}(A_{\text{gestures}}) = 4$ bits. If a system using the data
glove can recognize a more advanced set of gestures, each of which is comprised of one or two elementary gestures, the
advanced alphabet consists of $16 \times 16$ letters. The maximum entropy is increased to 8 bits. When we begin to study
the probability distributions of the elementary gestures and the composite gestures, this is very similar to Shannon’s
study of English letters and their compositions in data communication [33].

3.4 Input Device Utilization

As illustrated in Fig. 3(a), performing an HCI task involves two interrelated transformations: one is associated with an
input device alphabet and another with an input action alphabet; and one characterizes the resources used by the HCI
task, and another for the amount of knowledge delivered for the HCI task. The level of utilization of an input device

---

1Note: Repeating the same elementary gesture, e.g., “fist” + “fist”, is considered as one elementary gesture due to the ambiguity in recognition.

Manuscript submitted to ACM
can thus be measured by:

\[
DU = \frac{\text{Action Capacity}}{t \times \text{Device Bandwidth}} = \frac{C_{\text{act}}}{t \times W_{\text{dev}}}
\]

where \(t\) is the time (in unit: second) taken to perform the HCI task. In general, instead of using an accurate time \(t\) for each particular HCI task, one can use an average time \(t_{\text{avg}}\) estimated for a specific category of HCI tasks.

Using the examples in the above two subsections, we can estimate the DU for HCI tasks using radio buttons, check boxes, freehand paths, and gestures. Consider a set of four radio buttons with a uniform probability distribution, a portion of a display screen of \(128 \times 128\) pixels, and a simple 2D mouse with 2 on-off buttons and 100Hz sampling rate. Assume that the average input time is 2 seconds, we have:

\[
DU_{\text{radio}} = \frac{2}{2 \times 100 \times (7 + 7 + 1 + 1)} \text{bits/s} = \frac{2}{3200} \approx 0.06\%
\]

Following on the previous discussion on different probability distributions of an input action alphabet, we can easily observe that the more skewed distribution, the lower the action capacity \(C_{\text{act}}\) and thereby the lower the DU.

If the same mouse and the same portion of the screen device are used for a set of four check boxes with a uniform probability distribution and we assume that the average input time is 4 seconds, we have:

\[
DU_{\text{checkbox}} = \frac{2^4}{4 \times 100 \times (7 + 7 + 1 + 1)} \text{bits/s} = \frac{16}{6400} = 0.25\%
\]

Consider that the same 100 Hz mouse and the same portion of the screen device are used for drawing a freehand path with a uniform probability distribution for all possible paths. We assume that on average, a freehand path is drawn in 1 second, yielding 100 points along the path. The DU is thus:

\[
DU_{\text{freehand}} \approx \frac{(7 + 7) \text{bits} \times 100}{1 \times 100 \times (7 + 7 + 1 + 1)} \text{bits/s} = \frac{1400}{1600} = 87.5\%
\]

For gesture input using the aforementioned 200 Hz data glove, if an elementary gesture on average takes 2 seconds to be recognized with a reasonable certainty, the DU is:

\[
DU_{\text{gesture}} \approx \frac{4}{2 \times 10888.6} \text{bits/s} = \frac{4}{21777.2} \approx 0.02\%
\]

Some HCI tasks require additional display space or other resources for providing users with additional information (e.g., multiple-choice questions) and some do not (e.g., keyboard shortcuts). In the former cases, the extra resources should normally be included in the consideration of device bandwidth \(W_{\text{dev}}\). At the same time, the varying nature of the information should be included in the consideration of the input action alphabet. For instance, for a set of 10 different yes-no questions, there are two variables, one represents the options for the questions and one for the options of "yes" or "no". Hence the corresponding alphabet actually consists of 20 letters (e.g., \{"Q1-yes", "Q1-no", . . . , "Q10-yes", "Q10-no"\}) rather than just "yes" and "no". For more complicated additional information, such as different visualization images, we can extend the definitions in this work, e.g., by combining the (input) device utilization herein with the display space utilization defined in [9].

From the discussions in this section, we can observe that the quantitative measurement allows us to compare the capacity and efficiency of different input devices and HCI tasks in a sense that more or less correlates with our intuition in practice. However, there may also be an uncomfortable sense that the device utilization is typically poor for many input devices and HCI tasks. One cannot help wonder if this would support an argument about having less HCI.
4 COST-BENEFIT OF HCI

Chen and Jänicke raised a similar question about display space utilization [9] when they discovered that quantitatively the better utilization of the display space did not correlate to the better visual design. While they did identify the implication of visualization and interaction upon the mathematical proof of DPI as discussed in Section 1, the effectiveness and efficiency of visualization was not addressed until 2016 when Chen and Golan proposed their information-theoretic measure for analyzing the cost-benefit of data intelligence processes [8]. As HCI plays a valuable role in almost all nontrivial data intelligence workflows, we hereby use this measure to address the question about cost-benefit of HCI.

The cost-benefit measure by Chen and Golan considers three abstract measures that summarize a variety of factors that may influence the effectiveness and efficiency of a data intelligence workflow or individual machine- or human-centric processes in the workflow. Recently Chen et al. proposed an improvement to the original formula by replacing the unbounded divergence term with an bounded one [4, 10]. In this work, we adopt this new formula, which is:

\[
\frac{\text{Benefit}}{\text{Cost}} = \frac{\text{Alphabet Compression} - \text{Potential Distortion}}{\text{Cost}} = \frac{\mathcal{H}(Z_i) - \mathcal{H}(Z_{i+1}) - \mathcal{H}_{\text{max}}(Z_i) D_{\text{cs}}(Z'_i || Z_i)}{C_i}
\]

where \(Z_i = \{z_{i,1}, z_{i,2}, \ldots, z_{i,n_i}\}\) is the input alphabet to a process \(P_i\), and \(Z_i\) is associated with a probability distribution \(\Phi(Z_i) = \{\phi_1, \phi_2, \ldots, \phi_{n_i}\}\). \(Z_{i+1} = \{z_{i+1,1}, z_{i+1,2}, \ldots, z_{i+1,n_{i+1}}\}\) is the output alphabet of the process \(F_i\), and \(Z_{i+1}\) is associated with a distribution \(\Theta(Z_{i+1}) = \{\theta_1, \theta_2, \ldots, \theta_{n_{i+1}}\}\). \(Z'_i\) is an alphabet reconstructed by an inverse process \(G_i\), which may be considered as an approximation of \(F_i^{-1}\). \(Z'_i\) has the same letters as \(Z_i\) but is likely associated with a different probability distribution \(\Psi(Z'_i) = \{\psi_1, \psi_2, \ldots, \psi_{n_i}\}\).

Given an HCI process, which may represent the completion of an HCI task from start to finish, a micro-step during the execution of an HCI task, or macro-session comprising several HCI tasks, the measure first considers the transformation from an alphabet before the processing to the alphabet after the processing. As given in Eq. (3), this abstract measure is referred to as Alphabet Compression.

Consider the process \(P_i\) as a function, \(F_i: Z_i \rightarrow Z_{i+1}\), which consists of all actions from the point when a user starts executing a HCI task to the point when a computer stores the information about the input (in terms of the input action alphabet \(A_{\text{act}}\)) and is ready to forward this information to the subsequent computational processes, such as \(P_{i+1}, P_{i+2}\), and so on. In information theory, such a function is often referred to as a transformation from one alphabet to another. Alphabet compression measures the entropic difference between the two alphabets, \(\mathcal{H}(Z_i) - \mathcal{H}(Z_{i+1})\).

As discussed in Section 3, every HCI task is defined by an input action alphabet that captures the essence what a computer would like to know. The computer is uncertain before the transformation, and becomes certain after the transformation. The amount of uncertainty to be removed by a user’s interaction equals to the action capacity \(C_{\text{act}}\) of the input action alphabet \(A_{\text{act}}\). In terms of Eq. (3), we have \(C_{\text{act}} = \mathcal{H}(Z_i)\) since \(A_{\text{act}} = Z_i\).

At the end of the HCI task, the computer receives an answer from the user, the subsequent alphabet \(Z_{i+1}\) usually consists of only one letter (e.g., selecting a radio button). Therefore the entropy \(\mathcal{H}(Z_{i+1})\) is 0, and the alphabet compression \(\mathcal{H}(Z_i) - \mathcal{H}(Z_{i+1}) = \mathcal{H}(Z_i) = C_{\text{act}}\).

As illustrated in Fig. 3(b), alphabet compression measures an quantity about the forward mapping from \(Z_i\) to \(Z_{i+1}\). The more entropy is removed, the higher amount of alphabet compression, and hence the higher amount of benefit according to Eq. (3). If we did not have another measure to counter-balance alphabet compression, a computer randomly
chooses a ratio button or fails to recognize a gesture correctly would not have direct impact on the benefit of HCI. Therefore it is necessary to introduce the second abstract measure Potential Distortion, which is mathematically defined by the term \( H_{max}(Z_i) D_{cs}(Z'_i \| Z_i) \).

If a computer could intelligently think about the information provided by a user (e.g., a radio button or a gesture recognized from a video stream), the computer would not trust the information fully. The computer could doubt whether the user might have selected radio button C instead of B if the textual description were better, or a gesture of No. 3 might actually be No. 4. The potential distortion measures a quantity for the reverse mapping from \( Z_{i+1} \) to \( Z_i \). We use \( Z'_i \) to denote the alphabet resulting from this reverse mapping. \( Z'_i \) has the same set of letters as \( Z_i \), but usually a different probability distribution. If a computer can always detect and stores a user’s intended input correctly, the potential distortion \( D_{cs}(Z'_i \| Z_i) \) is 0. A high value of the potential distortion indicates a high level of inconsistency between the probability distributions of \( Z'_i \) and \( Z_i \). In information theory, there are many divergence measures for quantifying such inconsistency. Based on the theoretical and empirical evaluation by Chen et al. [4, 10], we use \( D_{cs}(Z'_i \| Z_i) \) in this work. As \( 0 \leq D_{cs}(Z'_i \| Z_i) \leq 1 \), the amount of potential distortion caused by a transformation is bounded by \([0, H_{max}]\).

Readers who are interested in the mathematical definitions of alphabet compression and potential distortion may also consult [8, 12] for further details.

The third abstract measure is the Cost of the process, which should ideally be a measurement of the energy consumed by a machine- or human-centric process. In practice, this is normally approximated by using time, a monetary quantity, or any other more obtainable measurement. For example, in HCI, we may use the average time, cognitive load, skill levels for a user to perform an HCI task, computational time, or monetary cost of computational resources for recognizing a human action. In fact, if we use device bandwidth as the cost while assuming that the computer always detects and stores the user’s input correctly, the cost-benefit measure is the same as the measure of input device utilization DU.

In fact, we have only examined the second transformation, \( F_i^a \), for performing a HCI task as depicted in Fig. 3(c) where there is less tangible and often unnoticeable first step. Before a user considers an input action alphabet \( A_{act} = Z_i \), the user has to take in and reason about various information that may affect an action of HCI. Collectively all possible variations of any information that may be considered for a HCI task are letters in an alphabet, denoted as \( Y_{i-1} \) in Fig. 3(c). Hence the first step of “taking in and reasoning about” is, in abstract, a transformation, \( F_i^a : Y_{i-1} \times Z_i \rightarrow Y_i \). As \( F_i^a \) takes place in a user’s mind, it is often unnoticeable. Broadly speaking, \( F_i^a \) may take in the following types of information.

**Explicit Prompt.** This includes any information that is purposely provided by a computer or a third party for the HCI task concerned, e.g., textual and visual prompts for radio buttons or checkboxes, audio questions asked prior to voice-activated commands, instructions from a trainer or a user manual to a trainee, and so forth.

**Situational Information.** This includes any information provided by a computer or an environment where interaction occurs. The information is not specifically, but can be used, for the HCI task. This may include the texts or drawings that a user is currently working on when the user issues a “save as” command, and the current sound or lighting quality in a video conference when the user issues a command to switch on or off the video stream.

**Soft Knowledge.** This includes any information that resides in the user’s mind, which can be called upon to support the HCI task. Tam et al. [39] considered two main types of soft knowledge: soft alphabets and soft models. The former encompasses factual knowledge that is not available as explicit prompts or situational information, e.g., the knowledge about keyboard shortcuts, the knowledge about the reliability for the computer to recognize a gesture or voice. The latter encompasses analytical knowledge that can be used to derive information for the HCI task dynamically. For
Fig. 4. A simple HCI task may be affected by three types of variables, which are collectively a very complex alphabet.

example, a user may assess the levels of risk associated to each radio button (or in general, each letter in \( \mathcal{A}_{\text{act}} \)). While the levels of risk are letters of a soft alphabet, the alphabet exists only after a soft model has been executed.

Fig. 4 shows an example of an HCI task. A user is editing a .tex file using a word processor (Microsoft Word) because it is easy to zoom in and out. After the user issues a “Ctrl-S” command, the computer displayed a pop-up window of 734×140 pixels, with a textual prompt. The input action alphabet \( \mathcal{A}_{\text{act}} \) has three multiple-choice buttons. Hence the maximal benefit that can be brought by the transformation \( F^b_i \) in Fig. 3(c) for this case is about 1.58 bits.

Meanwhile, the word processor may have different explicit prompts following a “Ctrl-S” command according to, e.g., the file modification status, the existence of a file with the same name, access permission, etc. A colleague may offer advice as to which button to choose. The display screen may show different situational information, e.g., different documents being edited, and different concurrent windows that may or may not be related the file being processed. The user may have the soft knowledge that a .tex file is a plain text file, the so-called “features” in the prompt cannot be processed by a \LaTeX{} compiler, the “help” button does not provide useful guidance to this particular way of using the word processor, and so on. As we can see that \( Y_{i-1} \) is not a simple alphabet and has a non-trivial amount of entropy, we can conclude that the two transformations, \( F_i^a \) and \( F_i^b \), together bring about benefit much more than 1.58 bits.

5 ESTIMATING COST-BENEFIT ANALYTICALLY

The cost-benefit metric described in Section 4 provides HCI with a mean for probabilistic characterization of complex phenomena. While it can be calculated from gathered data about every letter in an alphabet in some simple or highly controlled settings (e.g., see Section 6), it is more practical to estimate the three measures in real-world applications, e.g., for comparing different user interface designs or evaluating HCI facilities in a data intelligence workflow.\(^2\)

Let us first exemplify the estimation method by revisiting the channel selection scenario in Fig. 2 in Section 3. A coarse estimation can be made with an assumption that a user’s selection is always correct. In this case, the potential distortion in Eq. (3) is of 0 bits. The amount of alphabet compression thus equals to the action capacity \( C_{\text{act}} \). Meanwhile,\(^2\)

\(^2\)In thermodynamics, the notion of entropy provides a microscopic measure, reflecting the fundamental understanding about thermodynamic phenomena. It is typically estimated based on macroscopic quantities such as temperature, volume, and pressure that are more easily measurable.
from a usability perspective, the cost can be estimated based on the time, effort, or cognitive load required for selecting each option. For example, in Fig. 2(a), the top option \(a_1 \in A\) is the default selection, and requires only one [OK] action on the remote controller. The second option \(a_2\) requires a [▼] action followed by [OK], while the third option \(a_3\) requires three actions: [▼], [▼], and [OK]. If the time for each button action is estimated to take 2 seconds, the average cost for this HCI task is:

\[
\text{cost} = 2(p(a_1) + 2p(a_2) + 3p(a_3)) \quad \text{[unit: second]}
\]

Using the three example probability distributions for the input action alphabet \(A\) in Section 3, we can obtain,

\[
p(a_1) = p(a_2) = p(a_3) = 1/3 \implies \text{cost}(A_a) = 4
\]

\[
p(a_1) = 0.2, p(a_2) = 0.7, p(a_3) = 0.1 \implies \text{cost}(A_b) = 3.8
\]

\[
p(a_1) = 0.09, p(a_2) = 0.9, p(a_3) = 0.01 \implies \text{cost}(A_c) = 3.84
\]

Combining with the calculation of \(C_{\text{act}}\) in Eq. 2 in Section 3, we have the cost-benefit ratios for the three probability distributions are approximately \(A_a : 0.40 (= 1.58/4)\), \(A_b : 0.30 (= 1.16/3.8)\), and \(A_c : 0.13 (= 0.52/3.84)\) bits/s respectively. Hence for the skewed distribution associated with \(A_c\), the cost-benefit is very low.

During the design or evaluation of the TV system, a UX expert may discover that users select \(a_2\) “Select Channel” more frequently than the other two options. The UX expert can consider an alternative design by swapping the position of \(a_1\) and \(a_2\). With the changes of the corresponding probability distributions, the UX expert can value the improvement of the cost-benefit quantitatively, such as, \(A_b : 0.30 \nearrow 0.41\) and \(A_c : 0.13 \nearrow 0.23\).

A more detailed estimation may consider the factor that users may mistakenly press [OK] for the default option. For example, if in 20% cases, users are intended to select \(a_2\) but select the default \(a_1\) by mistake, there are both potential distortion and extra cost. In the case of \(A_b\), the reconstructed probability distribution is \(p'(a_1) = 0.4, p'(a_2) = 0.5, p'(a_3) = 0.1\). The potential distortion can be calculated as \(H_{\text{max}}D_{cs} \approx 1.58 \times 0.05 \approx 0.08\) bits. In the case of \(A_c\), the reconstructed probability distribution is \(p'(a_1) = 0.29, p'(a_2) = 0.7, p'(a_3) = 0.01\). The potential distortion is \(H_{\text{max}}D_{cs} \approx 1.58 \times 0.06 \approx 0.09\) bits. Let the extra time for showing detailed information about a TV show and going back to the original three options is 4 seconds. We can estimate that the extra time in the two cases are: \(20\% \times 4 = 0.8\) seconds on average. The cost benefit ratio will be reduced as: \(A_b : 0.30 \searrow 0.23\) and \(A_c : 0.13 \searrow 0.09\). If the mistakes were to reach 51% or more, the metric would return a negative value for \(A_c\).

Similarly, one may estimate the user’s effort as the cost by counting the steps needed to perform an action, such as reading the screen, looking at the remote control, and pressing a button. One may also weigh these steps differently based on pre-measured cognitive load for different types of elementary steps, which may be obtained, for instance, using electroencephalography (EEG) (e.g., [40]).

For the example in Fig. 2(b), it is easy to observe that the cost-benefit is always 0 since \(C_{\text{act}} = 0\) bits, though the cost for pressing [OK] on the remote control may not be considered high. This quantitative measure is consistent with what most UX experts would conclude qualitatively.

The estimation for the channel selection task does not consider any situational information or soft knowledge. When such variables are considered as part of an HCI task, as illustrated in Fig. 3(c), the amount of cost-benefit usually increase noticeably. For example, consider the \(\TeX\) example in Fig. 4. If the word processor on the left has 5 different pop-up windows in responses to a “save”, “save as”, or “Ctrl-S” command, each with 3 options, the input action alphabet has 15 letters. The maximum alphabet compression for the second process \(F_2\) in Fig. 3(c) is about 3.9 bits.
On the other hand, when given any one of 10 file types (e.g., .doc, .tex, .txt, .htm, etc.), the user has the knowledge about whether formatting styles matter. There are 10 binary variables or 10 bits of knowledge available. Consider conservatively that on average a user deletes or modifies 10 English letters independently before saving. The user knows whether it is critical to overwrite the existing file when a pop-up window asks for a confirmation. There are 10 nominal variables, each with some 26 valid values for English letters. As the entropy of English alphabet is about 4.7 bits [33], the total amount of knowledge available is about 47 bits. Without considering other factors (e.g., digits, symbols, etc.), we can conservatively estimate the amount of alphabet compression for the process $F_1$ in Fig. 3(c) is about $(10 + 47) - 3.9$ bits. Let us assume that selecting one of the three options takes 1 second. The cost-benefit for such a simple HCI task ($F_1 + F_2$) is at the scale of 57 bits/s.

Tam et al. [39] estimated the amount of human knowledge available to two interactive machine learning workflows. Both workflows were designed to build decision tree models, one for classifying facial expression in videos and other for classifying types of visualization images. They were curious by the facts that the interactive workflows resulted in more accurate classifiers than fully automated workflows. They estimated the amount of human knowledge available to the two workflows. Using the approach exemplified by the \LaTeX example, they identified 9 types of soft knowledge in the facial expression workflow and 8 types in the other. In both cases, there were several thousands of bits of knowledge available to the computational processes in the workflows.

6 MEASURING COST-BENEFIT EMPIRICALLY

As the cost-benefit ratio described in Section 4 is relatively new, there has been only one reported empirical study attempting to measure the three quantities in the formula. Kijmongkolchai et al. [20] conducted a study to measure the cost-benefit of three types of soft knowledge used during the visualization of time series data. This includes the knowledge about (i) the context (e.g., about an electrocardiogram but not weather temperature or stock market data), (ii) the pattern to be identified (e.g., slowly trending down), and (iii) the statistical measure to be matched with the time series plot.
The knowledge concerned can be considered as the transformation $F^a_i$ in Fig. 3(c), while the participants’ answers to the trial questions can be considered the transformation $F^b_i$. Kijmongkolchai et al. converted the conventional measures of accuracy and response time to that of benefit and cost in Eq. (3). In [20], Kijmongkolchai et al. described briefly the translation from (accuracy, response time) to (benefit, cost) with the support of a supplementary spreadsheet. Here we generalize and formalize their study, and present a conceptual design that can be used as a template for other empirical studies for detecting and measuring humans’ soft knowledge in HCI.

Consider a common design for a controlled experiment, in which an apparatus presents a stimulus to participants in each trial, poses a question or gives the input requirement, and asks them to make a decision or perform an HCI action. The participants’ action in response to the stimulus and input requirement is a human-centric form of data intelligence. Fig. 5(a) illustrates the workflow of such a trial.

A stimulus may comprise of textual, visual, audio, and other forms of data as the input to the process. Normally, one would consider that the alphabet, $S_{\text{stimulus}}$, contains only the stimuli designed for a trial or a set of trials for which participants’ responses can be aggregated. However, if the pre-designed stimuli are unknown to participants, one must consider that $S_{\text{stimulus}}$ consists of all possible stimuli that could be presented to the participants. For instance, the design of a study may involve only 64 pairs of colors and ask users to determine which is brighter. Since any pairing of two colors are possible, $S_{\text{stimulus}}$ actually consists of $N \times N$ letters where $N$ is the number of colors that can be shown on the study apparatus. On a 24-bit color monitor, $N = 2^{24}$ and $N \times N = 2^{48} \gg 64$. Hence the entropy of $S_{\text{stimulus}}$ is usually very high.

On the other hand, the participants’ inputs are commonly facilitated by multiple-choice buttons, radio buttons, or slide bars, which has a smaller alphabet $S_{\text{choice}}$. In some studies, more complicated inputs, e.g., spatial locations and text fields, are used, corresponding to large alphabets. Nevertheless, for any quantitative analysis, such complicated inputs will be aggregated to, or grouped into, a set of post-processed letters in a smaller alphabet $S'_{\text{choices}}$. It is not difficult to notice that $S_{\text{choice}}$ or $S'_{\text{choices}}$ is essentially an input action alphabet $A_{\text{act}}$. In the following discussion, we do not distinguish between $S_{\text{choice}}$ and $S'_{\text{choices}}$.

Once a participant has made a decision, only one letter in the alphabet $A_{\text{decision}}$ has the probability value 1, while the other letters are of probability 0. The $A_{\text{decision}}$ is thus of entropy 0 bits. However, after one merges all the repeated measures and responses from different participants into an alphabet $S_{\text{result}}$ (with the same letters of $A_{\text{decision}}$), the letters in $S_{\text{result}}$ are expected to be associated with difference numbers or frequencies of occurrence.

From the perspective of interaction, the alphabet $A_{\text{act}}$ has different probability distributions at different stages as illustrated in Fig. 5(a). Before and after the stimulus presentation stage, $A_{\text{act},1}$ has a ground truth for each trial, and thus one letter has the full probability 1. After the question stage, the letters in $A_{\text{act},2}$ are pretended to have an equal probability $1/\|A_{\text{act}}\|$. After the decision stage, only one letter in $A_{\text{act},3}$ is chosen, which thus has the full probability 1. After the aggregation stage, $A_{\text{act},4}$ has a probability distribution reflecting all repeated measures and all participants’ responses.

The humans’ soft knowledge used in the transformation from $S_{\text{stimulus}}$ to $S_{\text{choice}}$ and to $S_{\text{decision}}$ can be very complicated. The amount of alphabet compression can be huge. Nevertheless, the essence of any controlled experiment is to investigate one or a few aspects of this soft knowledge while restricting the variations of many other aspects. Here we refer one particular aspect under investigation as a *sub-model*, $S$, which may be a heuristic function for extracting a feature or factor from the stimulus or for retrieving a piece of information that is not in the stimulus.

Let us first examine a very simple “yes-no” trial designed to investigate if a sub-model $S$ has role to play in the transformation from $S_{\text{stimulus}}$ to $S_{\text{decision}}$. As illustrated in Fig. 5(b), at the beginning $A_{\text{act},1}$ has two letters $\{Y, N\}$. 

Manuscript submitted to ACM
Assuming that \( Y \) is the ground truth, the probabilities are \( p(Y) = 1, p(N) = 0 \). Note that in [20], Kijmongkolchais et al. introduced a tiny small value \( 0 < \epsilon < 1 \) to moderate \( p(Y) \) and \( p(N) \) in order to prevent the Kullback-Leibler divergence \( D_{KL} \) from handling the conditions of \( \log 0 \) and \( \div 0 \) because their calculation was based on the original formula for the cost-benefit measure by Chen and Golan [8]. Here we adopt the new formula by Chen and Sbert [4, 10], where the unbounded \( D_{KL} \) was replaced with the bounded \( D_{cs} \). Therefore, the awkward moderation by \( \epsilon \) is no longer necessary.

Any alphabet with two letters can potentially has a maximal entropy value of \( H_{\text{max}} = 1 \) bit. When the question, “yes” or “no”, is posed to each participant, \( A_{\text{act},2} \) is associated with the probabilities \( p(Y) = p(N) = 0.5 \). \( A_{\text{act},2} \) is thus of the maximal entropy 1 bit, indicating the maximal level of uncertainty.

When a decision is made by a participant, the probability distribution of \( A_{\text{act},3} \) is either \( p(Y) = 1, p(N) = 0 \) or \( p(Y) = 0, p(N) = 1 \). From an individual participant, the apparatus (e.g., the computer) receives an answer with certainty. \( A_{\text{act},3} \) is of minimal entropy 0 bits.

After all related responses are collected, \( A_{\text{act},4} \) has probabilities \( p(Y) = \alpha, p(N) = 1 - \alpha \). \( A_{\text{act},4} \) has the maximum amount of entropy of 1 bit, while that of \( A_{\text{act},4} \) is between 0 and 1 bits depending on \( \alpha \). If \( \alpha = 0.5 \) (e.g., random choices), the sub-model \( S \) offers no alphabet compression. If \( \alpha = 1 \) (i.e., all “yes” answers) or \( \alpha = 0 \) (i.e., all “no” answers), \( S \) enables 1 bit alphabet compression from \( A_{\text{act},2} \) to \( A_{\text{act},4} \). Without repeated measures, all participants individually achieve the same alphabet compression. We will discuss the case of repeated measures towards the end of this section.

Meanwhile, without repeated measures, the potential distortion has to be estimated using the collective results from all participants. As shown in Fig. 5(b), it is measured based on the reverse mapping from \( A_{\text{act},4} \) to \( A_{\text{act},1} \). As simple “yes-no” alphabet can be coded using binary codewords as \( A_{\text{act}} = \{N,Y\} = \{0,1\} \). \( A_{\text{act},1} \) and \( A_{\text{act},4} \) have the same set of letters but may have different probability distributions. As mentioned earlier, \( P(A_{\text{act},1}) = \{1,0,0,0\} \). If all participants have answered “yes”, we have \( \alpha = 1 \) and \( P(A_{\text{act},4}) = \{1,0,0,0\} \). Using the sub-formula for \( D_{cs} \) in Eq. 3, we have \( D_{cs} = 0 \). If all participants have answered “no”, \( \alpha = 0 \), \( P(A_{\text{act},4}) = \{0,1,0,0\} \), and \( D_{cs} = 1 \). Fig. 6(a) shows the trend of decreasing divergence \( D_{cs} \) when \( \alpha \) changes from 0 (i.e., all incorrect) to 1 (i.e., all correct).

For an empirical study designed to examine a sub-model at a slightly higher resolution, we can assign \( k > 1 \) bits to the input action alphabet \( A_{\text{act}} \). For example, a 3-bit alphabet \( A_{\text{act}} = \{a_1, a_2, \ldots, a_8\} \) can be labelled as \{000, 001, \ldots, 111\}. It is necessary to use all \( 2^k \) letters as choices in order to maximize the entropy of \( A_{\text{act},2} \) at the question stage. For examining the combined effects of several sub-models, we assign a bit string to each sub-model and then concatenate their bit strings together. For example, to study one 2-bit sub-model \( U \) and two 1-bit sub-models \( V \) and \( W \), we can have \( A_{\text{act}} = \{u_1v_1w_1, u_1v_1w_2, u_2v_2w_1, \ldots, u_4v_2w_2\} \) and can be labelled as \{0000, 0001, 0010, \ldots, 1111\}.\text{ }

Given an input action alphabet with \( n = 2^k \) letters (i.e., all possible answers in a trial), one assigns a ground truth in \( A_{\text{act},1} \), e.g., \( P(a_1) = 1 \) and \( P(a_i) = 0, i = 2, 3, \ldots, n \). In conjunction with a stimulus, one poses a question with \( n \) choices in \( A_{\text{act},2} \), which are pretended to have an equal probability of \( 1/n \). After an individual participant has answered the question, only one letter \( a_j \) in \( A_{\text{act},3} \) is selected, i.e., \( P(a_j) = 1 \) and \( P(a_i) = 0, i \in [1..n] \land i \neq j \). After collecting all related responses, the probability of each letter in \( A_{\text{act},4} \) is computed based on its frequency in participants’ responses. One can then convert the accuracy and response time to cost-benefit as:

\[
\text{benefit over cost} = \frac{H(A_{\text{act},2}) - H(A_{\text{act},3}) - H_{\text{max}}(A_{\text{act},2})D_{cs}(A_{\text{act},4}\|A_{\text{act},1})}{\text{average response time}}
\]

Using the experimental data in [20] as an example, we have a 3-bit input action alphabet for three sub-models (each with 1-bit resolution). Each of the eight possible answers in \( A_{\text{act}} \) is encoded by three bits \( b_1b_2b_3 \), where \( b_1 = 1 \) if the sub-model \( S \) functions correctly, and \( b_1 = 0 \) otherwise. In their experiment, the participants were presented with
eight optional answers that are randomly ordered. Hence, $\mathcal{H}(A_{act,2}) = 3$ bits and $\mathcal{H}(A_{act,3}) = 0$ bits. The alphabet compression for an individual is thus 3 bits. Their study obtained a set of accuracy data in terms of the percentages of eight possible answers in $A_{act,4}$. The values, which are depicted in Fig. 6(b), are as follows:

- 68.3% for letter 111 — $S_1$, $S_2$, $S_3$ are all correct.
- 10.7% for letter 110 — $S_1$ is correct.
- 10.3% for letter 101 — $S_2$, $S_3$ are correct.
- 4.6% for letter 100 — $S_1$, $S_3$ are correct.
- 2.5% for letter 011 — only $S_2$ is correct.
- 1.7% for letter 010 — only $S_3$ is correct.
- 1.1% for letter 001 — only $S_1$ is correct.
- 0.8% for letter 000 — $S_1$, $S_2$, $S_3$ are all incorrect.

The alphabet $A_{act,4}$ in fact represents the reconstruction of $A_{act,1}$ by the participants in response to $A_{act,2}$. As $A_{act,1}$ represents the ground truth, the potential distortion of $A_{act,4}$ is therefore measured against $A_{act,1}$, i.e., $\mathcal{H}_{\text{max}}(A_{act,4}∥A_{act,1})$. Using the above accuracy values for the probability distribution of $A_{act,4}$, we can obtain $\mathcal{H}_{\text{max}}(A_{act,4}∥A_{act,1})$ as 0.354 bits. Using the nominator of Eq. 4, the benefit is about 2.646 bits.

As suggested in Fig. 6(b), one may hypothesize how the benefit value would have changed if the study had returned different results. For example, one may imagine such changes by redistributing $x\%$ of the correct answers (labelled as 000) uniformly to the seven incorrect answers. Fig. 6(c) illustrates the impact of such hypothesized results. When $x\%$ increases from 0% to 100%, the benefit (solid purple line) decreases while the potential distortion (solid orange line) increases. Note that the benefit calculation in Eq. 4 is based on an assumption that the study is to simulate a scenario where a definite decision is made for each posed question. The probability distribution of $A_{act,4}$ will not be available to any succeeding processes.

Occasionally, we may use a study to simulate a group scenario where multiple group members offer their own independent answers, which are available to the succeeding processes as votes distribution in percentages. In the group scenario, the information of votes distribution conveys more uncertainty than a definite answer in the first scenario. There is thus less alphabet compression in the group scenario. For such a scenario, we replace the term...
The Value of Interaction in Data Intelligence

\[ \mathcal{H}(\lambda_{act,2}) - \mathcal{H}(\lambda_{act,3}) \text{ in Eq. 4 with } \mathcal{H}(\lambda_{act,2}) - \mathcal{H}(\lambda_{act,4}). \] As shown in Fig. 6(c), the benefit curve (dotted purple line) is lower than the solid purple curve for the scenario of a definite decision. From Fig. 6(c), one may notice that when some 26% (or more) correct answers are redistributed to incorrect answers, the benefit become negative. When about 90% correct answers are redistributed, the benefit reaches the minimum. This is because the entropy \( \mathcal{H}(\lambda_{act,4}) \) (dotted gray line) reaches its maximum at \( x\% \approx 82\% \), causing the alphabet compression (dotted cyan line) to reach its minimum.

The method of repeated measures is typically for dealing with difficulties in capturing definite inputs from each participant. The repeated measures for each participant are either aggregated first to yield a quasi-consistent measure or fused into the overall statistical measure involving all participants. As long as repeated measures are used for addressing such difficulties, one should still use Eq. 4. Only in a rare scenario where repeated measures are used to simulate an inconsistent decision process by an individual and the probability distribution of \( \lambda_{act,4} \) is calculated based on the repeated measures obtained from just one participant, one can use \( \mathcal{H}(\lambda_{act,4}) \) instead of \( \mathcal{H}(\lambda_{act,3}) \) in Eq. 4.

7 DATA INTELLIGENCE

In general, a data intelligence process is a transformation from some input data to some output data. The process can be a complex workflow or an infinitesimally small step in a workflow. The output data can be a final decision at the end of a workflow, or some intermediate data in a workflow. Hence, simple HCI processes, e.g., selecting a radio button, are in principle a data intelligence process. Of course, most people would relate the term “data intelligence processes” to more complex processes, and some might interpret the term as artificial intelligence (AI). As Chen and Golan showed in their work [8], the cost-benefit measure is applicable to both machine-centric and human-centric processes. In this section, we show that the information-theoretic measures, which were discussed in relatively “small” contexts in the previous sections, are also applicable to HCI in complex data intelligence workflows. We will use the development of classification models using machine learning (ML) as an example.

A typical classification model, \( M \), takes an input data object (e.g., an image, a sentence, a time series, etc.) and delivers a class label. In terms of information theory, \( M \) is a transformation from a data alphabet \( D \) to a class alphabet \( C \), such that \( D \) contains all possible input data objects in the context that \( M \) is used, and \( C \) contains all labels that \( M \) may generate. For example, \( D_a \) may contain all possible color images of \( W \times H \) pixels with an animal. \( C_a \) contains \( k \) labels for \( k \) types of animals. Alternatively, \( D_b \) may contain all possible color images of \( W \times H \) pixels, while \( C_b \) contains \( k + 1 \) labels for \( k \) animal classes plus one “unknown” class label. In terms of the number of letters, \( \|D_b\| \gg \|D_a\| \).

While \( M \) may likely be an automated process, the development of \( M \) is rarely automated, but involves a huge amount of HCI activities. We can estimate the cost-benefit of such HCI activities. In general, \( M \) is an instance among all possible functions that can be executed on a computer, which is referred to as the space of Turing Machine. In other words, an ML workflow is a transformation about an alphabet \( M \) that contains all possible functions in the space of Turing Machine. The transformation starts with \( M_0 \) where all functions are equally probable, gradually changes the associated probability distribution, and finally reaches an alphabet \( M_N \) where the trained model \( M \in M \) has the probability 1 and all other possible functions have the probability 0. The entropy of \( M_0 \) is \( \infty \) for a Turing Machine with infinite tape length, while the entropy of \( M_N \) is 0. The among of alphabet compression is \( \mathcal{H}(M_0) - \mathcal{H}(M_N) = \infty \).

As soon as an ML developer decides to use a specific algorithmic framework of ML (e.g., convolutional neural network (CNN) or decision tree (DT)), the initial alphabet \( M_0 \) is transformed to \( M_1 \), where the probability of many functions became 0. For example, the spaces of CNN and DT are known to be much smaller than the space of Turing Machine, so some functions, which can be written as a program in a conventional programming language, cannot be realized using a CNN or a DT. As illustrated in Fig. 7, if an ML developer selects “Supervised Learning”, it changes the probabilities of...
all functions in $\mathbb{M}_0$, resulting in $\mathbb{M}_1^{a}$ such that those functions that cannot be realized using supervised learning will have a probability 0, while those functions that can be found using supervised learning will become more probable than they were in $\mathbb{M}_0$. An apparently simple decision by the ML developer results in a huge amount alphabet compression.

When an ML developer further defines the template structure of an ML model, hypo-parameters, the feature set, many functions in $\mathbb{M}_1$ become impossible or less probable. Let $\mathbb{M}_2$ encompasses all remaining functions with their probabilities. The transformation from $\mathbb{M}_1$ to $\mathbb{M}_2$ involves many HCI activities, and the amount of alphabet compression is usually huge, hence the benefit of HCI can be huge.

For supervised learning, the labels assigned to data objects are typically entered by using HCI. For a classification process, each data label makes a small contribution, usually much less than $\log_2 l$ bits due to the redundant, dependent, and conflicting information among the labels, to the alphabet compression from $\mathbb{M}_2$ that contains numerous candidate functions (i.e., models) to a specific model $M^a$. Therefore it is necessary to have a lot of data labels.

It is rare for an ML workflow to deliver a model with only one training process. The evaluation of a trained model by ML developers usually leads to a decision to invoke another iteration with modified hypo-parameters and sometimes the structure and feature selection. Since each iteration delivers a model $M^a$ in alphabet $\mathbb{M}_i^{a}$, models from iterations 1, 2, …, $i$ (i.e., $M_1^a, M_2^a, \ldots, M_i^a$) are trained and evaluated, one of which is more likely to be selected than others. Hence the probability distribution of $\mathbb{M}_i^{a}$ changes with increasing $i$ during the iterative training processes. In an ML workflow, there are usually many iterations, lasting for months. There are thus a lot of HCI activities.
For active learning, there are also HCI activities during the training processes. Through HCI, ML developers are able to influence the evolution of the model space, \( L \), dynamically within a training process. Typically, the goal of HCI was to speed up the convergence (e.g., alphabet compression) by selecting more helpful training data.

The terms such as “Self-supervised Learning” or “Unsupervised Learning” may give an impression that these techniques do not rely on human input. It is mostly true that these techniques demand less HCI activities as it is not necessary to use HCI to label thousands or millions of data objects. However, through a smaller number of HCI activities, ML developers provide the ML processes with much critical knowledge. As shown in Fig. 7, in the case of “Reinforcement Learning”, a fitness function needs to be defined, and in the case of “Self-supervised Learning”, two or more interaction functions are needed. Some unsupervised learning techniques require a distance function, which demands a similar level of alphabet compression by using HCI. Consider an alphabet \( \mathbb{F} \) that includes all candidate functions, it would have numerous letters. Finding a letter \( f \in \mathbb{F} \) to be a fitness function or interaction function is a transformation that enables a noticeable amount of alphabet compression. For example, if there were 1 million candidate functions in \( \mathbb{F} \), the HCI activities for entering a particular function \( f \) into the computer would enable nearly 20 bits alphabet compression. During the training process, the function \( f \) would be used again and again in all iterations. If the training process converges, \( f \) contributes to the alphabet compression from \( M_0 \) to \( M_N \).

As mentioned at the beginning of this section, we consider the development of classification models as an example of data intelligence workflows. While the ML techniques under the four algorithmic frameworks on the left of Fig. 7 produce models that can perform classification tasks, unsupervised learning techniques (under the fifth framework on the right) typically produce models that group the data objects in the training data into \( k \) clusters. Such clustering models are not classification models yet, but can be transformed to classification models using HCI.

For example, it is common to project multivariate data objects onto a 2D scatter plot, where data objects are depicted as 2D points and color-coded according to their corresponding cluster labels. With such a scatter plot, an ML developer can divide the 2D space into \( k \) partitions, e.g., by drawing polylines. The partitioning effort usually demands some complex human decisions as the clusters often overlap with each other in the projected space. Hence the HCI activities inject human knowledge that the computer do not have. Since there are numerous ways of divide the 2D space, HCI enables a huge amount of alphabet compression in the last step, “Interpretation”, of the unsupervised learning workflow on the right of Fig. 7. For example, consider that the 2D space is at the \( 1024 \times 1024 = 2^{20} \) pixel resolution, and there are \( k = 8 = 2^3 \) clusters. Theoretically, any one of the \( 2^{20} \) pixels may be associated with any one of the clusters. The alphabet \( M_{N-1}^e \) could have up to \( 3 \times 2^{20} \) bits of uncertainty (the maximal entropy).

If an ML observes the visual patterns of the 8 clusters using the scatter plot and draws polylines to partition the 2D space, there are \( 2^{20} \) options for each point on these polylines. In other words, the input action alphabet for each input (see Section 3) has \( 2^{20} \) letters. If the ML developer managed to divide the 2D space into 8 partitions with \( 32 = 2^5 \) inputs of 2D points, the computer would have received \( 2^{25} \) bits of information. With the information, the computer would be able to produce a classification model such that when any new data object is projected onto one of 8 partitions, it is classified with the same cluster label for that partition. In producing \( M_N^e \), the computer would have used the \( 2^{25} \) bits of human inputs to achieve alphabet compression of \( 3 \times 2^{20} \) bits.

8 CONCLUSIONS

In this paper, we have shown that information-theoretic measures can provide a mathematical approach to evaluate the cost-benefit of input actions. In conjunction with the previous discourse on the cost-benefit of visualization by Chen and Golan [8], we can now appreciate the value of the two-way HCI quantitatively. We have shown the applicability of
this approach to basic input actions (e.g., selecting a radio button) as well as to complex input actions (e.g., defining a function in ML). The measures can be estimated analytically and measured empirically.

The information-theoretic approach presented in this paper is not a replacement for but an addition to the existing toolbox for supporting the design and evaluation of HCI devices, interfaces, and systems. Because this approach allows us to examine the benefit of HCI to computers, i.e., from a perspective different from the commonly adopted focuses on the benefits to human users, it offers a new tool complementary to the existing qualitative and quantitative methods.

With estimated or measured quantitative values of HCI, we can appreciate more the necessity of HCI, especially in data intelligence workflows. We showed an analysis of a few ML workflows corresponding to widely-used algorithmic frameworks. The quantitative analysis confirms numerous qualitative arguments about humans’ role in ML and AI, e.g., in the conversations in the HCAI Google Group [36]. To study the value of HCI is not in any way an attempt to forestall the advancement of technologies such as data mining, ML, and AI. On the contrary, such research can help us understand better the transformation from human knowledge to computational models, and help us develop better automated processes to be used in data intelligence workflows. As shown in an ontological map by Sacha et al. [31], many steps in ML workflows have benefited, or can benefit, from visualization and interaction. It is indeed not the time to reduce HCI in data intelligence, but to design and provide more cost-beneficial HCI.

In this paper, we have touched a few topics in a relatively broad spectrum of HCI in order to demonstrate the generality and applicability of the several information-theoretic measures discussed. Like all mathematical approaches, the proposed information-theoretic approach no doubt needs further experimentation, refinement, enrichment, and improvement before it can become a deployable tool in practical applications. Hopefully it will become part of the long-term endeavor of the HCI community.

REFERENCES

[1] B. Bach, E. Freeman, A. Abdul-Rahman, C. Turkay, S. Khan, Y. Fan, and M. Chen. 2023. Dashboard design patterns. IEEE Transactions on Visualization and Computer Graphics 29, 1 (2023).
[2] R. G. Bias and D. J. Mayhew. 2005. Cost Justifying Usability (2nd ed.). Elsevier, Oxford, UK.
[3] P. Cairns and A. L. Cox (Eds.). 2008. Research Methods for Human-Computer Interaction. Cambridge University Press.
[4] M. Chen, A. Abdul-Rahman, D. Silver, and M. Shert. 2022. A bounded measure for estimating the benefit of visualization (Part II): case studies and empirical evaluation. Entropy 24, 2 (2022), 282.
[5] M. Chen and D. S. Ebert. 2019. An ontological framework for supporting the design and evaluation of visual analytics systems. Computer Graphics Forum 38, 3 (2019), 131–144.
[6] M. Chen, M. Feixas, I. Viola, A. Bardera, H.-W. Shen, and M. Shert. 2016. Information Theory Tools for Visualization. A K Peters/CRC Press.
[7] M. Chen, K. Gaiither, N. W. John, and B. McCann. 2019. Cost-benefit analysis of visualization in virtual environments. IEEE Transactions on Visualization and Computer Graphics 25, 1 (2019), 32–42.
[8] M. Chen and A. Golan. 2016. What May Visualization Processes Optimize? IEEE Transactions on Visualization and Computer Graphics 22, 12 (2016), 2619–2632.
[9] M. Chen and H. Janicke. 2010. An Information-theoretic Framework for Visualization. IEEE Transactions on Visualization and Computer Graphics 16, 6 (2010), 1206–1215.
[10] M. Chen and M. Shert. 2022. A bounded measure for estimating the benefit of visualization (Part I): theoretical discourse and conceptual evaluation. Entropy 24, 2 (2022), 228.
The Value of Interaction in Data Intelligence

[20] N. Kijmongkolchai, A. Abdul-Rahman, and M. Chen. 2017. Empirically measuring soft knowledge in visualization. Computer Graphics Forum 36, 3 (2017), 73–85.
[21] J. Lazar, J. H. Feng, and H. Hochheiser. 2010. Research Methods in Human-Computer Interaction. Wiley.
[22] A. Light, P. J. Wild, A. Dearden, and M. J. Muller. 2005. Quality, Value(s) and Choice: Exploring deeper outcomes for HCI products. In Proc. CHI ’05 (Workshops). 2124–2125.
[23] I. S. MacKenzie. 2013. Human-Computer Interaction: An Empirical Research Perspective. Morgan Kaufmann.
[24] K. Norman and J. Kirakowski (Eds.). 2018. The Wiley Handbook of Human Computer Interaction. Wiley-Blackwell.
[25] A. Oulasvirta, P. O. Kristensson, X. Bi, and A. Howes (Eds.). 2018. Computational Interaction. Oxford University Press.
[26] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. 1994. Human-Computer Interaction: Concepts And Design. Addison Wesley.
[27] J. Preece, H. Sharp, and Y. Rogers. 2015. Interaction Design: Beyond Human-Computer Interaction (4th ed.). John Wiley.
[28] H. C. Purchase. 2012. Experimental Human-Computer Interaction: A Practical Guide with Visual Examples. Cambridge University Press.
[29] A. Rotondo and N. Freier. 2010. The Problem of Defining Values: A Lack of Common Ground Between Industry & Academia?. In Proc. CHI ’2010 (Work-in-Progress). 4183–4188.
[30] E. Rydow, R. Borgo, H. Fang, T. Torsney-Weir, B. Swallow, T. Forphrey, C. Turkay, and M. Chen. 2023. Development and evaluation of two approaches of visual sensitivity analysis to support epidemiological modelling. IEEE Transactions on Visualization and Computer Graphics 29, 1 (2023).
[31] D. Sacha, M. Kraus, D. A. Keim, and M. Chen. 2019. VIS4ML: An ontology for visual analytics assisted machine learning. IEEE Transactions on Visualization and Computer Graphics 25, 1 (2019).
[32] C. E. Shannon. 1948. A mathematical theory of communication. Bell System Technical Journal 27 (1948), 379–423.
[33] C. E. Shannon. 1951. Prediction and Entropy of Printed English. Bell System Technical Journal 30 (1951), 50–64.
[34] K. Shilton. 2018. Values and Ethics in Human-Computer Interaction. Foundations and Trends in Human-Computer Interaction 12, 2 (2018), 107–171.
[35] B. Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proc. IEEE Symposium on Visual Languages. 336–343.
[36] Ben Shneiderman. Accessed in September 2022. Human-Centered-AI. https://groups.google.com/g/human-centered-ai.
[37] B. Shneiderman, C. Plaisant, M. Cohen, and S. Jacobs. 2010. Designing the user interface: strategies for effective human-computer interaction. Pearson.
[38] W. Smith, G. Wadley, S. Webber, B. Ploderer, and R. Lederman. 2014. Unbounding the Interaction Design Problem: the Contribution of HCI in Three Interventions for Well-being. In Proc. OzCHI ’14. 392–395.
[39] G.-K. L. Tam, V. Kothari, and M. Chen. 2017. An analysis of machine- and human-analytics in classification. IEEE Transactions on Visualization and Computer Graphics 23, 1 (2017), 71–80.
[40] D. S. Tan and A. Nijholt (Eds.). 2010. Brain-Computer Interfaces: Applying our Minds to Human-Computer Interaction. Springer.
[41] Z. Ye and M. Chen. 2023. Visualizing ensemble predictions of music mood. IEEE Transactions on Visualization and Computer Graphics 29, 1 (2023).

Manuscript submitted to ACM