Anthropomorphised Algorithms.

Are humanisations of machine learning algorithms suitable for shedding light on the social impacts of artificial intelligence?

Joachim Heller*

2. 8. 2024

Abstract

Anwendungen maschinellen Lernens zu betrachten als Algorithmen, die eigenständige, aber absolut undurchschaubare gesellschaftliche Akteure sind, ohne die damit verbundenen theoretischen Annahmen aus einer technischen Perspektive zu überprüfen, verstellt den Blick auf den eigenen Anteil an der Ohmachtsempfindung gegenüber «computationaler Autorität». Einseitige Erklärungsansätze entfalten so auch keine politisch-emanzipierende Wirkung in der Medienbildung. Diese These wird vorliegend anhand der Rezeption eines Aufsatzes entwickelt, in dem die technische Perspektive als ungeeignet für die Ermittlung der Verstärkungswirkung von Diskriminierungstendenzen verworfen wird. Zur Kontrastierung dieser Auffassung wird die Kybernetik als methodischer Rahmen vorgeschlagen, der eine transparentere Terminologie bietet, um auch medienpädagogisch im Grundschulunterreicht anschließen zu können.

One's own contribution to the feeling of powerlessness in the face of «computational authority» is obscured, if applications of machine learning are considered as algorithms that are independent, inscrutable social actors, without questioning the associated theoretical assumptions from a technical perspective. Biased explanatory approaches therefore do not have a politically emancipatory effect in media education. This thesis is developed on the basis of the reception of an essay in which the technical perspective is rejected as unsuitable for determining the reinforcing effect of discrimination tendencies. To contrast this view, cybernetics is proposed as a methodological framework that offers a more transparent terminology in order to be able to incorporate media education in primary school teaching.

---

*Humboldt-Universität zu Berlin, M.Ed. student in lateral entry to primary school teaching, contact: joachim.heller@hu-berlin.de – Anthropomorphised Algorithms ©2024 by Joachim Heller is licensed under CC BY 4.0 ©①.

Contents

List of Figures

1 Introduction

There are two perspectives for defining the term 'algorithm': one from a mathematical-technical perspective and one from everyday language. The latter encourages analogies that are given social scientific meaning and are intended to be compatible with media education. In particular, the image of a quasi-living entity is legitimised, whose cultural influence is supposedly made transparent through these analogies. In the following, an exposition written within this paradigm will be used to demonstrate how this strategy itself obscures the understanding of algorithms and, when applied in media education, can amplify inequalities. In section 2 the framework set up by the exposition's author is first considered, then in 3 his fallacies are discussed and in 4 their effects are traced.

2 Habitual-posthumanistic-neomaterialistic realm

Waldmann assumes that algorithms in software applications of machine learning are irreducibly opaque (Waldmann 2024, pp. 1, 3). Notably, the posthumanist-neo-materialistic

position based on *Barad* seems to have resonated with the author himself: he uses its terminology throughout (see Barad 2003). That position suggests a definition of 'algorithm' in which a recognisable self-consciousness (cf. Hegel 2014, pp. 145–147) emerges, which, in conjunction with a habitually focused position, is a socialisable *relatively independent actor* with *unpredictable power to act* (Waldmann 2024, p. 2). From a non-posthumanist perspective, such an idea can be viewed as humanisation. The peculiar thing is that the anthropomorphisation of hardware components mentioned by *von Foerster* in 1970, (von Foerster 2002, pp. 171–173), are obviously completely ignored in the argument and the humanisation takes place solely on the software level. This may initially be a 'Cartesian cut', with which the mind (the algorithm) *is considered* separated from the matter (the hardware) (Primas 1994, p. 610). In addition, the 'agential cut' is then supposed to have the effect that performatively, i. e. during and through presentation of the posthumanist observation, a previously non-existent 'relatum' is *created*, here the artificial subject 'algorithm', which is related to the analysis and independently *materialises itself* in the world thus reconfigured; in this case, the analysis is the 'apparatus' with which the cut is made 'intra-actively' and the components of the relationship only gain meaning for the analysts – perhaps also the hardware as a "mechanical apparatus" (constructed according to Waldmann 2024, S. 8, 11–13 m.w.N., Everth and Gurney 2022, p. 16; Barad 2003, p. 815).

Based on this, Waldmann would like to use a *posthumanist skepticism* as an alternative or complement to a sociologically influenced media pedagogical *triad* (Waldmann 2024, pp. 14–19), that enables a position with which the opacity of algorithms (established as a given) can be adequately addressed in order to accept their output not doubtfully but skeptically as a possible truth with contingency expectations and not as certainty (ibid., pp. 19–21).

Cova sees a need for more substantial definitions of the digital in order to be able to more precisely target social scientific instruments for determining its social impact; he suggests paying more attention to the aspect of the generation of discrete (digital) and continuous (analog) signals and their transformation from one form to another and to reconsider structuralism for this (Cova 2016). Structuralism involves the recognition of connecting patterns, but has been criticized for not answering questions about the possibilities for changing power structures (Graf 2010, pp. 212–213).

Waldmann assumes a *structural definition* of 'algorithm' (Waldmann 2024, p. 3), which he suspects of unreflectively reproducing the habitus and power structures of *white *cis men with academic backgrounds* (ibid., p. 5). As far as the focus on power structures is to be sharpened, structuralism seems difficult to convey.

Nevertheless, it is necessary to find another theoretical approach, since Waldmann's analysis is technically incorrect and pedagogically problematic. His approach not only

does not offer any possibility of changing power structures, but also has the destructive potential of inadvertently supporting what is to be avoided.

Another approach is Luhmannian systems theory. In view of the frequent criticism that it neglects humans (cf. for example Luhmann 2013, p. 35, note 47), it should at least offer the possibility for convergences from a skeptical *post*-humanist position itself. It is also just as stimulating and eclectic an epistemological project as Barad's quantum physical analogies.

The sociological arsenal of concepts shaped by systems theory leads back to cybernetics. The latter's coherent terminology for analysing and describing the behaviour of existing and hypothetical machines of any complexity (admittedly rejected by Barad in favour of her concept of "apparatus", Barad 2003, p. 816) in a wide variety of contexts, including social systems (for an overview, see Ashby 1957, pp. 2–6), has been absorbed into sociology and numerous other disciplines (in mathematics it remained as *control theory*, cf. Russell and Norvig 2010, p. 15). «Artificial intelligence», which is also an application field of cybernetics (see for example Anderson 2024, p. 16; Sieniutycz 2020, pp. 1–2), can be described more accurately and comprehensibly with the widely well-tried cybernetic conceptual system, from which terms such as 'determined machine', 'black box', 'variety' in relation to control systems, etc. come. This also enables a connection to computer science. Since basic arithmetic is sufficient for their elementary understanding (Ashby 1957, p. v), their knowledge also offers potential for media education, especially for educational offerings at primary schools.

In the following, its terminology will first be incorporated into the explanation of Waldmann's erroneous reasoning.

3 Misconceptions

Waldmann concludes from an asserted fact:

> "Entstehung und Effekte algorithmischer Ungleicheit [können] **nicht** allein durch bewusst herbeigeführte Entscheidungen, die technische Artefakte betreffen, beobachtet und erklärt werden. [emphasis added]" (Waldmann 2024, p. 4)

> *The emergence and effects of algorithmic inequality can**not** be observed and explained solely by consciously made decisions concerning technical artifacts.*

to an ought intended by him:

> "[D]ie figurativen, diskursiven und relational-performativen Wirkungsweisen von Algorithmen [**sind**] genauer im Hinblick auf hierarchisierende und ausschliessende Logiken zu analysieren **und nicht** von handlungszentrierten Debatten über Transparenz und Kontrolle. [emphasis added]" (ibid., p. 4)

> *The figurative, discursive and relational-performative effects of algorithms **are to be** analysed more closely with regard to hierarchical and exclusionary logics **and not** "of action-centered debates about transparency and control".*

The chosen subjunction (material conditional) is in itself simply arbitrary: the entire statement is only false if the first claim is true, but the second partial statement is untenable. The equivalent would be to assert adjunctively: "*At least* with one of them, the relevant here can be observed and explained, *or* with the other it should be analysed." Now, a normative sentence cannot be true and so Waldmann is not even saying anything gratuitous at this point, but nothing at all. His further conclusions based on this therefore hang up in the air.

3.1 Algorithms

There is no "structural definition" of 'algorithm'. The only source referred to by Waldmann, from which he believes he has taken this information as a consensus in computer science and which at the same time, in line with his normative specification, leads to Barad, is *Burke*, who again overemphasizes the intellectual movement of *structured programming* (Burke 2019, p. 2), whose concept the latter mistakenly considers "more-or-less" as a precursor to other programming concepts such as object-oriented and functional programming (ibid., p. 3), and he characterises the dogma of the separation of instructions ("code") and data as a reality-forming agential cut (ibid., pp. 3–4), as well as the summary of algorithms in program libraries (ibid., p. 8). The magical thinking of "a disempowered user outside an opaque decision-making system" (ibid., p. 9) from which an alternative anthropomorphised, "imprecise" algorithm concept arises (ibid., p. 10), forms the basis for non-technical discourses on the cultural impact of algorithms (ibid., pp. 10–12): the *systemic definition* in Waldmanns sense (Waldmann 2024, p. 3). In certain cases (for example didactic ones), this simplification can be informally useful even in technical discourse (Burke 2019, p. 10).

However, the opacity of algorithms and their effects usually becomes clearer to the extent that the willingness to deal with them more precisely *also* as technical artifacts increases. In terms of programming, there are basically three forms of representation of algorithms. They can be formulated using everyday language, for example as shown in Figure 1.

Algorithm S (*Selection sampling technique*). To select n records at random from a set of N, where $0 < n \leq N$.

S1. [Initialize.] Set $t \leftarrow 0$, $m \leftarrow 0$. (During this algorithm, m represents the number of records selected so far, and t is the total number of input records that we have dealt with.)

S2. [Generate U.] Generate a random number U, uniformly distributed between zero and one.

S3. [Test.] If $(N - t)U \geq n - m$, go to step S5.

S4. [Select.] Select the next record for the sample, and increase m and t by 1. If $m < n$, go to step S2; otherwise the sample is complete and the algorithm terminates.

S5. [Skip.] Skip the next record (do not include it in the sample), increase t by 1, and go back to step S2. ▮

Figure 1: Generally formulated algorithm (Knuth 1998, p. 142)

This representation can then be translated more or less literally into source code as a second form of representation (figure 2) which is converted into machine language and becomes executable code.

```
(defun algorithm-s (n max) ; max is N in Knuth's algorithm
  (let (seen              ; t in Knuth's algorithm
        selected          ; m in Knuth's algorithm
        u                 ; U in Knuth's algorithm
        (records ()))     ; the list where we save the records selected
    (tagbody
     s1
       (setf seen 0)
       (setf selected 0)
     s2
       (setf u (random 1.0))
     s3
       (when (>= (* (- max seen) u) (- n selected)) (go s5))
     s4
       (push seen records)
       (incf selected)
       (incf seen)
       (if (< selected n)
           (go s2)
           (return-from algorithm-s (nreverse records)))
     s5
       (incf seen)
       (go s2))))
```

Figure 2: Literally translated, executable algorithm (Seibel 2011b, p. 251)

The third form of representation can be considered as transposition into a version corresponding to the structure of this language – as shown in figure 3.

```lisp
(defun s-n-creator (n)
  (let ((sample (make-array n :initial-element nil))
        (i 0))
    (lambda (item)
      (if (<= (incf i) n)
          (setf (aref sample (1- i)) item)
          (when (< (random i) n)
            (setf (aref sample (random n)) item)))
      sample)))

(defun algorithm-s ()
  (let ((*random-state* (make-random-state t))
        (frequency (make-array '(10) :initial-element 0)))
    (loop repeat 100000
          for s-of-n = (s-n-creator 3)
          do (flet ((s-of-n (item)
                      (funcall s-of-n item)))
               (map nil
                    (lambda (i)
                      (incf (aref frequency i)))
                    (loop for i from 0 below 9
                          do (s-of-n i)
                          finally (return (s-of-n 9))))))
    frequency))
```

Figure 3: Idiomatically transposed executable algorithm (Collective 2024)

Depending on the means of abstraction, the algorithm can appear completely detached in the equivalent third representation (figure 4).

```lisp
(defun algorithm-s (n max)
  (loop for seen from 0
        when (< (* (- max seen) (random 1.0)) n)
          collect seen and do (decf n)
        until (zerop n)))
```

Figure 4: Maximally simplified algorithm (Seibel 2011b, p. 252)

What is formulated as a compact loop in this programming language (Common Lisp) is translated by the automatic system in the background for the computer into jump instructions that correspond to the original formulation of the algorithm. The structure of this algorithm fits well into the internal workings of computers and ends with a result (as intended by Knuth 1998, p. v).

In a procedural *machine-oriented* programming language (see e.g. Knuth 2005, p. iv) the translation of the algorithm can look like in figure 5.

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

struct s_env {
    unsigned int n, i;
    size_t size;
    void *sample;
};

void s_of_n_init(struct s_env *s_env, size_t size, unsigned int n)
{
    s_env->i = 0;
    s_env->n = n;
    s_env->size = size;
    s_env->sample = malloc(n * size);
}

void sample_set_i(struct s_env *s_env, unsigned int i, void *item)
{
    memcpy(s_env->sample + i * s_env->size, item, s_env->size);
}

void *s_of_n(struct s_env *s_env, void *item)
{
    s_env->i++;
    if (s_env->i <= s_env->n)
        sample_set_i(s_env, s_env->i - 1, item);
    else if ((rand() % s_env->i) < s_env->n)
        sample_set_i(s_env, rand() % s_env->n, item);
    return s_env->sample;
}

int *test(unsigned int n, int *items_set, unsigned int num_items)
{
    int i;
    struct s_env s_env;
    s_of_n_init(&s_env, sizeof(items_set[0]), n);
    for (i = 0; i < num_items; i++) {
        s_of_n(&s_env, (void *) &items_set[i]);
    }
    return (int *)s_env.sample;
}

int main()
{
    unsigned int i, j;
    unsigned int n = 3;
    unsigned int num_items = 10;
    unsigned int *frequencies;
    int *items_set;
    srand(time(NULL));
    items_set = malloc(num_items * sizeof(int));
    frequencies = malloc(num_items * sizeof(int));
    for (i = 0; i < num_items; i++) {
        items_set[i] = i;
        frequencies[i] = 0;
    }
    for (i = 0; i < 100000; i++) {
        int *res = test(n, items_set, num_items);
        for (j = 0; j < n; j++) {
            frequencies[res[j]]++;
        }
        free(res);
    }
    for (i = 0; i < num_items; i++) {
        printf(" %d", frequencies[i]);
    }
    puts("");
    return 0;
}
```

Figure 5: Algorithm translated into low(er)-level programming language C (Collective 2024)

8

In the last defined procedure `main`, the sub-procedure `test` is called, which calls previously defined procedures, which in turn call previously defined procedures. The users of such languages have significantly fewer automatisms at their disposal, so that a data structure suitable for the solution is also user-defined at the beginning. On a small scale, the hierarchical and sequential structure of such very concrete languages becomes clear and it is measurable how complex more far-reaching problem solutions can become. The concept of structured programming is intended for these languages – precisely *without* jumps (Dijkstra 1970, 1968; Criticism of the generalization: Harvey 1997, pp. 233–238), which of course are also implemented in the later machine code.

A language like the one demonstrated in figures 2 to 4 hides more of the technical peculiarities of computers and thus offers other ways of dealing with complexity, which add the equivalent *bottom-up design* (Graham 1993, pp. v–vi) in computer science to the *top-down programming principle* (Waldmann 2024, p. 2).

They and their relatives also manage without the dogmatic separation of data and code at the syntactical level. Data and code are structurally identical there, which enables further abstraction tools to reduce complexity (about the property initially Mooers and Deutsch 1965, p. 232).

Functional programming in languages such as the one demonstrated is a programming style based on mathematical function composition to exploit *recursive* processes and increase the reliabilty of complex program systems, especially in parallel processes (Turner 2013; Bauer 2004, pp. 247–248), what is particularly accessible to cybernetic observation (von Foerster 2015, p. 73). Recursive processes, on the other hand, are avoided as much as possible in the less automated machine-oriented languages due to the more complex memory management (Bauer 2004, p. 240). However, recursive procedures are not unstructured or irregular (opposite opinion apparently Waldmann 2024, p. 3). They are generalised loop constructions that repeat the rules for working through tree-like, i.e. equally recursive data structures for each individual element until a stop condition is reached (Domkin 2021, p. 159; on the special case of tail-call recursion/iteration G. Teschl and S. Teschl 2013, p. 221; Abelson, G. J. Sussman, and J. Sussman 1996, pp. 45–47).

There is no such thing as *the* object-oriented programming. Nonetheless, the different models have in common that the user-defined data-structures are associated with code in a more dynamic way than in structured programming (introductory Seibel 2011a, pp. 189–191). These models were developed *to overcome* the limitations of structured programming at even higher levels of complexity

All of these methods for organizing program source code support different problem-solving strategies in which algorithms are represented differently. Inspite of that, algorithms remain what they were before digital computers existed (Cormen et al. 2009, p. xv): A step-by-step procedure for solving a precisely described computational prob-

lem that, if followed correctly, leads to a correct result (Cormen et al. 2009, pp. 5–6) – which is already true for ancient methods of Euklid or Heron (cf. e.g. G. Teschl and S. Teschl 2013, pp. 94–95, 177; Landa 1969, pp. 22–24).

To use them reliably, you only need to know how they behave. Written division, for example, does not require any understanding of numbers (Padberg and Benz 2021, pp. 226–227). Procedures are *always* used as *black boxes* (Ashby 1957, p. 86), which, as *deterministic machines* (ibid., p. 24), show *reliable* transformations from appropriate input to output values (ibid., pp. 11–16, 86–87). A good behaviour description also gives programmers an indication of the side effects, but otherwise the internal processes are usually irrelevant unless correctness or efficiency are put to the test.

The insights Waldmann believes he has gained from Burke about the nature of algorithms are not tenable in this respect. Nevertheless, he rejects the technical perspective as trivial; its knowledge would lead to *biased results* and to *cultural phantasms* (Waldmann 2024, pp. 3, 5).

3.2 Raising awareness of decision options that affect technical artifacts

Complex software solutions are multiply coupled systems (Ashby 1957, pp. 48–53), in which repeated transformations take place (ibid., pp. 16–23), Markov chains may be implemented for machine learning (ibid., pp. 165–173), other methods based on Bayesian probability statistics are used (Downey 2012, pp. 3–6; Russell and Norvig 2010, pp. 495–499) and numerous other approaches – but no *truth* (Waldmann 2024, p. 20). Two things become obvious: (1) The principles on which machine learning applications are based are explainable. (2) Complex chains with multiple transformations cannot be described in detail by the transition of the respective input values from one component to the next. However, it remains undeniable that the input of a self-learning system is transformed into an output, as well as that this transformation takes place on a rational basis, namely, as a rule, statistical procedures (also Burke 2019, pp. 8–9).

Using the representative example of a study on the use of software to exploit precariously employed delivery service employees in false self-employment by the actors in an ideal-typical anti-labour law start-up company (Waldmann 2024, p. 9), Waldmann attempts to demonstrate the posthumanist-neomaterialist analysis tool presented and to show the inevitable opacity of algorithms in applications of artificial intelligence.

For this purpose, on the one hand, a *machine habitus* is assumed, which represents the habitual characteristics of the people involved in the development and the automated learning processes, which is referred to as 'primary' and 'secondary' *machine socialisation*, from which path dependencies arise, which ultimately form a *practical sense of judgment* of the machines, albeit one that is difficult to predict (ibid., pp. 5–6). The sociological interpretation of the fictitious, representative case study should then show

that the software originally used *constitutes a techno-social field* in terms of its habitus (Waldmann 2024, p. 10). The laws of that field are incomprehensible to all people involved, but the software realizes symmetries at the level of the habitual participation of all actors, which then also includes the machine (ibid., p. 11). On the other hand, the neo-materialist interpretation is intended to show, in contrast to this, but also in addition, that machines create reality through *algorithms as agential cuts* and are thus also independent participants in the discourse on inequalities that is interwoven with matter (ibid., pp. 13–14), which preserves the opacity of the algorithms.

Nevertheless, Waldmann himself gives the example of a correct output hypothesis in this context by drawing attention to the options for action in connection with the technical artifact, where actual courier drivers were able to specifically and successfully influence the output of their orderer's software in their favour (ibid., p. 18).

Another example of methodical behavioural analysis is provided by two computer scientists from the group of "BIPoC" (which is surprisingly underrepresented in Waldmann's sources ibid., p. 4), with a research focus on bias and artificial intelligence (cf. Buolamwini 2016, 2017), who have the competence to adequately observe, describe and analyse the transformations `Input→Output` and thus the *reproducible* behaviour of applications of complex statistical methods of machine learning. Due to the necessity of using automatic facial recognition for law enforcement purposes in the USA and other countries, *Buolamwini* and *Gebru* have comprehensibly and reproducibly identified the inaccuracies in the facial recognition of three software products *using an appropriately prepared data source* and have thus *clearly* demonstrated that, as an undesirable side effect, the statistical weightings coded at the time of their investigation do *not* lead to correct results, and they can provide recommendations for action to improve the product (Buolamwini and Gebru 2018, p. 12). In fact, the focus in programming and analysis is generally more on the *data* used and generated and their structure than on the algorithms, which, however, does not mean a separation in the sense of structured programming (other opinion Burke 2019, pp. 8–9), but rather allows the complexity to be shifted from the program code to the transparent data (cf. Domkin 2021, pp. 29–30). Even the hidden ratings are data whose tendencies can be measured by the error rate. The analysis by Buolamwini/Gebru also allows for interpretations: It may say something about the proportion of *white *cis men with academic backgrounds* at IBM if the error rate of their product is 0.0% when it comes to facial recognition of light-skinned women (Buolamwini and Gebru 2018, p. 10). Above all, however, Waldmann's paradigm is opposed: technical analysis is *particularly* suitable for making algorithmically generated inequality transparent and providing effective recommendations for action to correct errors – and is therefore *relevant* (currently e.g. Sabbatini and Calegari 2024; older contribution to further ethical problems, from a more humanistic perspective: Van der Loos 2014).

In the following, an implication of Waldmann's position is explored in more detail and why it can be considered highly problematic in media education, at least in primary schools.

4 Responsibility

The humanisation of algorithms follows, as seen in Burke, from feelings of powerlessness and a lack of willingness to acquire agency – based on an "argument from ignorance" (Walton, Reed, and Macagno 2013, p. 327): *If these algorithms were comprehensible, I could understand them. But I cannot.* or a fallacy of generalisation: *I cannot understand it because nobody can.*

This may have something to do with *movement*: in the old *Common Law* of England, the ancient legal concept that a thing could become *deodand* survived until the 19th century (Holmes 1991, pp. 7–10). By 'thing' is meant everything that is used as a 'dependent' thing by 'fully accepted' people: tools, means of transport, farm animals and, earlier, slaves. If objects were brought to 'life' by movement or animals passed through and this caused significant damage, they were in certain cases *forfeited* and *fell to God* (*deo-dandum*) or to the crown or the public or even the injured party as compensation for the misfortune that had occurred; it was about *no-fault liability* in the case of damage that was not directly attributable (e.g. Waldmann 2024, S. 7 unten), but which was nevertheless intended to affect *people* closely associated with the event to a clearly defined extent (Holmes 1991, pp. 24–26). The owners were not directly obliged to pay compensation and were not subject to physical punishment or condemnation, but they lost this part of their property. Punishment and ostracism could instead affect the forfeited item, including in the form of its disposal.

This traditional strategy to solve the problem seemed increasingly absurd (House of Commons 1846), which involved pretending that a sailing ship would start to have a life of its own if (and only if) it broke free from its mooring without any further intervention, i.e. as if by itself, and was set in motion by the river current and damaged a jetty, causing injury to a person (Holmes 1991, pp. 22–23, 26–27).

There are more legally appropriate solutions than the detour via the anthropomorphisation of a 'scapegoat'. In liability law, this example shows the increasing rationalisation of a fundamental social problem through history. With a *habitual and powerful, intraactive reality decomposition apparatus*, the opposite path is taken to the idea of a thing coming to life through (recursive loop) movement (see also Turkle 2010, pp. 8–10). Symptomatically, a work slave, *Roboti* (*Karel Čapek*), could become a *deodand* again. A construct of the *machine habitus* or a general, posthumanist-neomaterialist-based skepticism towards the output does not lead to the cause: the enabling of *algorithmic authority* (Waldmann 2024, p. 11; with clearly human responsibility: Bryson 2010).

From a pedagogical point of view, it seems to me to be more promising if, on the one hand, data and its algorithmic processing and generation are made tangible and describable in order to be able to critically understand the transformation `Input`→`Output`. And if, on the other hand, awareness of personal responsibility is raised by which decision-making *tools* (computer *aided* design/planning/decision-making) are effective in which way, and also to what extent individuals permit to submit to the communication processes carried out with them or to exploit them for their own benefit and to the detriment of others.

5 Conclusion and outlook

Two provoking and interesting concepts, the construction of a machine habitus and agential cuts, are used to try to sharpen a blunt idea of algorithms without allowing for technical expertise that could dampen one's own normative reservations and prove preferred ideas and the arguments based on them to be unsustainable.

This creates an "irreducible opacity" that is not necessarily inherent in algorithms in the context of machine learning, but is created by speaking about *the effect* of the technology in an external way without engaging with its nature. This is the reproduction and amplification of the self-limiting powerlessness from which the everyday term 'algorithm' arose.

The possibilities of cybernetic observation were suggested as an alternative approach to encourage an examination of the origins of the established concepts. Without error-prone translation into and back translation from less established terminology they put the technical side into words that can be taken up in social science disciplines with understanding and may potentially enable a transdisciplinary dialogue. The path to this dialogue can already be prepared with the content-rich cybernetic terms, appropriately didactically reduced, in primary school education and linked to political education for recognizing and shaping power relations.

References

Abelson, Harold, Gerald Jay Sussman, and Julie Sussman (1996). *Structure and Interpretation of Computer Programs*. 2nd ed. Cambridge, MA, USA: MIT Press. URL: https://web.mit.edu/6.001/6.037/sicp.pdf (visited on 01/21/2017).

Anderson, Marc M. (2024). "AI as Philosophical Ideology: A Critical Look Back at John McCarthy's Program". In: *Philosphy & Technology* 37.44. URL: https://doi.org/10.1007/s13347-024-00731-1 (visited on 07/14/2024).

Ashby, W. Ross (1957). *An Introduction to Cybernetics*. 2nd ed. London, GBR: Chapman and Hall Ltd. URL: `https://philarchive.org/go.pl?id=ASHAIT-6&proxyId=&u=https%3A%2F%2Fphilpapers.org%2Farchive%2FASHAIT-6.pdf` (visited on 07/14/2024).

Barad, Karen (2003). "Posthumanist Performativity: Toward an Understanding of How Matter Comes to Matter". In: *Signs: Journal of Women in Culture and Society* 28.3, Gender and Science: New Issues, pp. 801–831. JSTOR: `10.1086/345321`. URL: `https://www.jstor.org/stable/10.1086/345321` (visited on 07/22/2024).

Bauer, Friedrich L. (2004). "Die Algol-Verschwörung". In: *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*. Ed. by Hans Dieter Heilige. Berlin/Heidelberg, GER: Springer-Verlag, pp. 237–254. URL: `https://link.springer.com/chapter/10.1007/978-3-642-18631-8_10` (visited on 12/03/2023).

Bryson, Joanna (2010). "Robots Should Be Slaves". In: *Close Engagements with Artificial Companions: Key Social, Psychological, Ethical and Design Issues*. Ed. by Yorick Willks. Natural Language Processing. Amsterdam, NED: John Benjamins Publishing Company, pp. 63–74. DOI: `10.1075/nlp.8.11bry`. URL: `https://www.researchgate.net/publication/250333956_Robots_Should_Be_Slaves` (visited on 07/26/2024).

Buolamwini, Joy (2016). *Unmasking Bias*. URL: `https://medium.com/mit-media-lab/the-algorithmic-justice-league-3cc4131c5148#.lh6ftfaoa` (visited on 09/08/2023).

– (2017). "Gender Shades: Intersectional Phenotypic and Demographic Evaluation of Face Datasets and Gender Classifiers". PhD thesis. Cambridge, MA, USA: Massachusetts Institute of Technology. URL: `https://dspace.mit.edu/bitstream/handle/1721.1/114068/1026503582-MIT.pdf` (visited on 09/08/2023).

Buolamwini, Joy and Timnit Gebru (2018). "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". In: *Proceedings of Machine Learning Research*. Ed. by Sorelle A. Friedler and Christo Wilson. Vol. 81. New York, USA: PMLR, pp. 1–15. URL: `http://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf` (visited on 07/24/2024).

Burke, Adam (2019). "Occluded Algorithms". In: *Big Data & Society* July–December, pp. 1–15. URL: `https://doi.org/10.1177/2053951719858743` (visited on 07/20/2024).

Collective, Rosetta (2024). "Knuth's Algorithm S". In: *Rosetta Code*. URL: `https://rosettacode.org/wiki/Knuth%27s_algorithm_S` (visited on 07/24/2024).

Cormen, Thomas H. et al. (2009). *Introduction to Algorithms*. 3rd ed. Cambridge, MA, USA: MIT Press.

Cova, Victor (2016). "Digitality, Analogicity, and Computation". In: *Theorizing the Contemporary, Fieldsights* March, 24. URL: `https://culanth.org/fieldsights/digitality-analogicity-and-computation` (visited on 05/30/2024).

Dijkstra, Edgar W. (1968). "Go To Statement Considered Harmful". In: *Communications of the ACM* 11.3, pp. 147–148. URL: `https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf` (visited on 07/15/2024).

– (1970). *Notes on Structured Programming.* Tech. rep. T.H.-Report 70-WSK-03. Eindhoven, NED: Technische Hogeschool Eindhoven, pp. 147–148. URL: `https://www.cs.utexas.edu/~EWD/ewd02xx/EWD249.PDF` (visited on 07/15/2024).

Domkin, Vsevolod (2021). *Programming Algorithms in Lisp: Writing Efficient Programs with Examples in ANSI Common Lisp.* New York, USA: Apress.

Downey, Allen B. (2012). *Think Bayes: Bayesian Statistics in Python.* Needham, MA, USA: Green Tea Press. URL: `https://www.greenteapress.com/thinkbayes/thinkbayes.pdf` (visited on 03/09/2024).

Everth, Thomas and Laura Gurney (2022). "Emergent Realities: Diffracting Barad within a Quantum-Realist Ontology of Matter and Politics". In: *European Journal for Philosophy of Science* 12.51. URL: `https://doi.org/10.1007/s13194-022-00476-8` (visited on 07/23/2024).

Graf, Erich Otto (2010). "Strukturalismus". In: *Wissenschaftstheorie.* Ed. by Detlef Horster and Wolfgang Jantzen. Vol. 1. Behinderung, Bildung, Partizipation Enzyklopädisches Handbuch der Behindertenpädagogik. Stuttgart, GER: Kohlhammer, pp. 211–214.

Graham, Paul (1993). *On Lisp.* Hoboken, New Jersey, USA: Prentice Hall. URL: `https://paulgraham.com/onlisptext.html` (visited on 11/02/2017).

Harvey, Brian (1997). *Computer Science Logo Style: Volume 1. Symbolic Computing.* 2nd ed. Cambridge, MA, USA: MIT Press. URL: `http://people.eecs.berkeley.edu/~bh/pdf/v1ch13.pdf` (visited on 08/18/2015).

Hegel, Georg Wilhelm Friedrich (2014). *Phänomenologie des Geistes.* 13th ed. Vol. 3. Werke. Frankfurt am Main, GER: Suhrkamp Verlag.

Holmes, Oliver Wendell jr. (1991). *The Common Law (Reprint of the 1st Edition of 1881).* New York, USA: Dover Publications, Inc.

House of Commons (1846). "Deodands Abolition (No. 2) Bill." In: *House of Commons Debate (HC Deb) 11 August 1846,* Vol. 88, cc. 624-7. URL: `https://api.parliament.uk/historic-hansard/commons/1846/aug/11/deodands-abolition-no-2-bill` (visited on 07/21/2024).

Knuth, Donald E. (1998). *Seminumerical Algorithms.* 3rd ed. Vol. 2. The Art of Computer Programming. Reading, MA, USA et. al.: Addision-Wesley. URL: `https://archive.org/details/artofcomputerpro0000knut/mode/2up` (visited on 07/03/2024).

– (2005). *MMIX: A RISC Computer for the New Millennium.* Vol. 1. Fascicle 1. The Art of Computer Programming. Upper Saddle River, New Jersey, USA: Addision-Wesley.

Landa, Lev N. (1969). *Algorithmierung im Unterricht. Autorisierte Übersetzung und Bearbeitung von Elske Däbritz (Original: Moskau 1966)*. Berlin: Volk und Wissen Volkseigener Verlag.

Luhmann, Niklas (2013). *Das Recht der Gesellschaft*. 6th ed. Frankfurt a.M., Deutschland: Suhrkamp Verlag.

Mooers, Calvin N. and L. Peter Deutsch (1965). "Programming Languages for Non-Numeric Processing—1: TRAC, a Text Handling Language". In: *ACM '65: Proceedings of the 1965 20th national conference* 4. August 1965, pp. 229–246. URL: `https://doi.org/10.1145/800197.806048` (visited on 07/24/2024).

Padberg, Friedhelm and Christiane Benz (2021). *Didaktik der Arithmetik. fundiert, vielseitig, praxisnah*. 5th ed. Berlin, Deutschland: Springer Spektrum.

Primas, Hans (1994). "Realism and Quantum Mechanics". In: *Logic, Methodology and Philosophy of Science*. Ed. by Dag Prawitz, Brian Skyrms, and Dag Westerståhl. Amsterdam, NED: Elsevier Science, pp. 609–631. URL: `https://philsci-archive.pitt.edu/951/1/Realism%26QuantumMechanics.pdf` (visited on 07/23/2024).

Russell, Stuart J. and Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. 3rd ed. Upper Saddle River, New Jersey, USA: Pearson. URL: `https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf` (visited on 12/20/2019).

Sabbatini, Federico and Roberta Calegari (2024). "On the Evaluation of the Symbolic Knowledge Extracted from Black Boxes". In: *AI and Ethics* 4, pp. 65–74. URL: `https://doi.org/10.1007/s43681-023-00406-1` (visited on 07/27/2024).

Seibel, Peter (2011a). "Object Reorientation: Generic Functions". In: *Practical Common Lisp*. New York, USA: Apress. URL: `https://gigamonkeys.com/book/object-reorientation-generic-functions` (visited on 02/11/2019).

– (2011b). "The Special Operators". In: *Practical Common Lisp*. New York, USA: Apress. URL: `https://gigamonkeys.com/book/the-special-operators` (visited on 02/11/2019).

Sieniutycz, Stanisław (2020). *Complexity and Complex Thermo-Economic Systems*. Amsterdam, NED: Elsevier. URL: `https://doi.org/10.1016/B978-0-12-818594-0.00001-5` (visited on 07/24/2024).

Teschl, Gerald and Susann Teschl (2013). *Mathematik für Informatiker. Band 1. Diskrete Mathematik und Lineare Algebra*. 4th ed. Berlin/Heidelberg, Deutschland: Springer Vieweg.

Turkle, Sherry (2010). "In Good Company? On the Threshold of Robotic Companions". In: *Close Engagements with Artificial Companions: Key Social, Psychological, Ethical and Design Issues*. Ed. by Yorick Willks. Natural Language Processing. Amsterdam, NED: John Benjamins Publishing Company, pp. 3–10. DOI: `10.1075/nlp.8.11bry`. URL: `http://web.mit.edu/people/sturkle/pdfsforstwebpage/Turkle_In%20Good%20Company.pdf` (visited on 07/30/2024).

Turner, David A. (2013). "Some History of Functional Programming Languages (Invited Talk)". In: *Trends in Functional Programming: 13th International Symposium, TFP 2012 St. Andrews, UK, June 12-14, 2012 Revised Selected Papers.* Ed. by Hans Wolfgang Loidl and Ricardo Peña. Heidelberg, GER/New York, USA/Dordrecht, NED/London, GBR: Springer, pp. 1–21. URL: https://link.springer.com/chapter/10.1007/978-3-642-40447-4_1 (visited on 03/10/2024).

Van der Loos, H. F. Machiel (2014). "Sherry Turkle. Alone Together: Why We Expect More from Technology and Less from Each Other. Basic Books, New York, 2012. ISBN-13: 978-0465031467 (Book Review)". In: *Science and Engineering Ethics* 20, pp. 5–6. URL: https://doi.org/10.1007/s11948-014-9524-1 (visited on 07/28/2024).

von Foerster, Heinz (2002). "Thoughts and Notes on Cognition". In: *Understanding Understanding: Essays on Cybernetics and Cognition.* Ed. by Heinz von Foerster. New York, USA: Springer-Verlag, pp. 169–189. URL: https://www.alice.id.tue.nl/references/foerster-2003.pdf (visited on 07/20/2024).

– (2015). "Kybernetik". In: *Wissen und Gewissen. Versuch einer Brücke.* Ed. by Siegfried J. Schmidt. 9th ed. Frankfurt am Main, GER: Suhrkamp Verlag, pp. 72–76.

Waldmann, Maximilian (2024). "Abwerten, Aussortieren, Separieren. Algorithmische Ungleichheit als neuartiges Gegenstandsfeld soziologischer, neomaterialistischer und medienbildungstheoretischer Positionen". In: *MedienPädagogik. Zeitschrift für Theorie und Praxis der Medienbildung* Themenheft 61, pp. 1–23. URL: https://doi.org/10.21240/mpaed/61/2024.06.10.X (visited on 07/20/2024).

Walton, Douglas, Chris Reed, and Fabrizio Macagno (2013). *Argumentation Schemes.* 6th ed. New York, USA: Oxford University Press.