

Vermenschlichte Algorithmen


Sind Anthropomorphisierungen von Algorithmen maschinellen Lernens allein geeignet, die gesellschaftliche Wirkung künstlicher Intelligenz zu erhellen?

Joachim Heller*

2. 8. 2024

Zusammenfassung

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 Ohnmachtsempfindung 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 Grundschulunterricht anschließen zu können.

**Humboldt-Universität zu Berlin*, Studierender zum M.Ed. im Quereinstieg zum Grunschullehramt, Kontakt: joachim.heller@hu-berlin.de – Vermenschlichte Algorithmen ©2024 by Joachim Heller is licensed under CC BY 4.0 .

Inhaltsverzeichnis

1	Einleitung	2
2	Habituell-posthumanistisch-neomaterialistisches Universum	3
3	Fehlannahmen	5
3.1	Algorithmen	5
3.2	Bewusstmachung von Entscheidungsmöglichkeiten, die technische Artefakte betreffen	10
4	Verantwortlichkeit	12
5	Schlussfolgerung und Ausblick	13

Abbildungsverzeichnis

1	Allgemein formulierter Algorithmus (Knuth 1998, S. 142)	6
2	Wörtlich übersetzter, ausführbarer Algorithmus (Seibel 2011b, S. 251) . .	7
3	Idiomatisch transponierter ausführbarer Algorithmus (Rosetta-Kollektiv 2024)	7
4	Maximal vereinfachter Algorithmus (Seibel 2011b, S. 252)	8
5	In maschinennahe Programmiersprache übersetzter Algorithmus (Rosetta-Kollektiv 2024)	9

1 Einleitung

Es gibt zwei Perspektiven, um den Begriff von ‹Algorithmus› zu definieren: Eine mathematisch-technische und eine alltagssprachliche. Letztere regt zu Analogien an, die sozialwissenschaftlich mit Bedeutung aufgeladen werden und in der Medienpädagogik anschlussfähig sein sollen. Insbesondere wird das Bild einer quasi-lebendigen Entität legitimiert, deren kultureller Einfluss über diese Analogien transparent werde. Im Folgenden wird an einer in diesem Paradigma verfassten Erörterung herausgearbeitet, wie diese Strategie selbst das Verständnis von Algorithmen verdunkelt und bei Anwendung in der Medienbildung Ungleichheiten verstärken kann. In 2 wird dafür zunächst der vom Autor aufgespannte Rahmen betrachtet, um sodann in 3 seine Irrtümer auseinanderzusetzen und in 4 deren Wirkung nachzuspüren.

2 Habituell-posthumanistisch-neomaterialistisches Universum

Waldmann geht davon aus, dass Algorithmen in Software-Anwendungen maschinellen Lernens undurchschaubar sind und ihre Intransparenz auch nicht vermindert werden kann (Waldmann 2024, S. 1, 3). Dabei scheint insbesondere die auf Barad gestützte posthumanistisch-neomaterialistische Position beim Autor selbst zu verfangen, deren Terminologie durchgehend verwendet wird (dazu Barad 2003). Mit ihr wird eine Begriffsbestimmung von *Algorithmus* nahegelegt, in der ein anerkennungsfähiges Selbstbewusstsein (vgl. Hegel 2014, S. 145–147) zum Vorschein kommt, das in Verbindung mit einer habituell fokussierten Position ein:e sozialisierbare:r „relativ eigenständige[:r] Akteur[:in]“ mit „unvorhersehbare[r] Handlungsmacht“ sei (Waldmann 2024, S. 2). Aus nicht-posthumanistischer Perspektive lässt sich eine solche Vorstellung als Vermenschlichung betrachten. Eigentümlich ist dabei, dass entgegen der etwa von Foerster im Jahr 1970 beschriebenen Anthropomorphisierungen von Hardware-Komponenten (Foerster 2002, S. 171–173), diese offensichtlich argumentativ vollständig ausgeblendet werden und die Vermenschlichung allein auf Software-Ebene geschieht. Das ist gegebenenfalls zunächst ein *«kartesischer Schnitt»*, mit dem Geist (der Algorithmus) von der Materie (die Hardware) losgelöst *betrachtet* wird (Primas 1994, S. 610). Mit *«agentiellem Schnitt»* wird darüber hinaus wohl bewirkt, dass performativ, also während der und durch die Darlegung der posthumanistischen Betrachtung, ein zuvor nicht existentes *«Relatum» hervorgebracht* wird, hier das künstliche Subjekt *Algorithmus*, das in Relation steht zur Analyse und in der so rekonfigurierten Welt eigenständig materialisiert; die Analyse ist in diesem Fall der *«Apparatus»*, mit welchem der Schnitt *«intra-aktiv»* vollzogen wird und die Komponenten der Beziehung erst Bedeutung erlangen, für die Analyst:innen vielleicht auch die Hardware als „maschineller Apparat“ (konstruiert nach Waldmann 2024, S. 8, 11–13 m.w.N., Everth und Gurney 2022, S. 16; Barad 2003, S. 815).

Darauf fußend möchte Waldmann alternativ oder ergänzend zu einer soziologisch geprägten medienpädagogischen „Triade“ (Waldmann 2024, S. 14–19) anschließen mit einer „posthuman[istisch]en Skepsis“, die eine Position ermögliche, mit welcher der als gegeben festgestellten Undurchsichtigkeit von Algorithmen angemessen begegnet werden könne, um deren „Output“ nicht zweifelnd aber skeptisch als mögliche Wahrheit mit Kontingenzerwartung entgegenzunehmen und nicht als Gewissheit (ebd., S. 19–21).

Cova sieht Bedarf an robusteren Definitionen des Digitalen, um sozialwissenschaftliche Instrumente zur Bestimmung dessen gesellschaftlicher Wirkung genauer justieren zu können; er legt nahe, sich stärker dem Aspekt der Erzeugung diskreter (digitaler) und kontinuierlicher (analoger) Signale sowie ihrer Wandlung von der einen in die andere Form zuzuwenden und dazu wieder den Strukturalismus in Betracht zu ziehen (Cova 2016). Strukturalismus geht mit der Erkenntnis verbindender Muster einher, steht je-

doch in der Kritik bei Anfragen zu Änderungsmöglichkeiten von Machtstrukturen zu versagen (Graf 2010, S. 212–213).

Waldmann sieht eine „strukturelle Definition“ von ›Algorithmus‹ (Waldmann 2024, S. 3), die er verdächtigt, unreflektiert den Habitus und die Herrschaftsstrukturen „weisse[r] *Cis-Männer mit akademischem Hintergrund“ zu reproduzieren (ebd., S. 5). Soweit der Blick auf Machtstrukturen geschärft werden soll, scheint der Strukturalismus schwer zu vermitteln.

Gleichwohl ist es notwendig, einen weiteren theoretischen Ansatz zu finden, da Waldmanns Analyse technisch falsch ist und pädagogisch problematisch. Sein Denkansatz stellt nicht nur keine Änderungsmöglichkeiten von Machtstrukturen in Aussicht sondern birgt darüber hinaus das destruktive Potenzial, unbeabsichtigt zu unterstützen, was vermieden werden soll.

Ein weiterer Ansatz ist die Systemtheorie luhmann'scher Prägung. Hinsichtlich der vielfachen Kritik, diese vernachlässige den Menschen (vgl. etwa Luhmann 2013, S. 35, Anm. 47) müsste sie aus selbst skeptischer *post*-humanistischer Position zumindest die Möglichkeit für Konvergenzen bieten. Sie ist zudem ein ebenso anregendes, eklektizistisches erkenntnistheoretisches Projekt, wie sich die quantenphysikalischen Analogien Barads präsentieren.

Das soziologisch-systemtheoretische Begriffsarsenal führt zurück zur Kybernetik. Deren kohärente Terminologie zur Analyse und Beschreibung des Verhaltens existenter und hypothetischer Maschinen beliebiger Komplexität (von Barad zugunsten ihres „Apparatus“-Begriffs freilich verworfen, Barad 2003, S. 816) in mannigfaltigsten Kontexten, auch etwa Gesellschaftssysteme (zum Überblick Ashby 1957, S. 2–6), ist in der Soziologie und zahlreichen weiteren Disziplinen aufgegangen (in der Mathematik verblieb sie als *Kontrolltheorie*, vgl. Russell und Norvig 2010, S. 15). Mit dem daher weitläufig erprobten kybernetischen Begriffssystem, dem etwa ›determinierte Maschine‹, ›Black Box‹, ›Vielfalt‹ in Bezug auf Regelsysteme etc. entstammen, kann «*die künstliche Intelligenz*», ebenfalls ein Anwendungsfeld der Kybernetik (dazu etwa Anderson 2024, S. 16; Sieniutycz 2020, S. 1–2), zutreffender und auch nachvollziehbarer beschrieben werden. Das ermöglicht zugleich den Anschluss an die Informatik. Da für ihr grundlegendes Verständnis die Grundrechenarten ausreichen (Ashby 1957, S. v), birgt ihre Kenntnis auch Potential für die Medienpädagogik, insbesondere für Bildungsangebote in der Grundschule.

Im Folgenden wird ihre Terminologie zunächst einfließen in die Darlegung von Waldmanns Irrtümern.

3 Fehlannahmen

Waldmann schließt von einem behaupteten Sein:

„Entstehung und Effekte algorithmischer Ungleichheit [können] **nicht** allein durch bewusst herbeigeführte Entscheidungen, die technische Artefakte betreffen, beobachtet und erklärt werden.“ (Waldmann 2024, S. 4, Hervorh. J.H.)

auf ein von ihm intendiertes Sollen:

„[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“. (ebd., S. 4, Hervorh. J.H.)

Die gewählte Subjunktion ist an sich schlicht beliebig: Die gesamte Aussage ist nur dann falsch, wenn die erste Behauptung zwar zutrifft, die zweite Teilaussage jedoch unhaltbar ist. Äquivalent wäre adjunktiv zu behaupten: «Mindestens *mit dem einen lässt sich das hier relevante beobachten und erklären* oder *mit dem anderen soll es analysiert werden*». Nun ist ein normativer Satz nicht wahrheitsfähig und daher wird im Grunde von Waldmann an dieser Stelle insgesamt nicht einmal beliebiges, sondern überhaupt nichts ausgesagt. Seine weiteren darauf aufbauenden Schlussfolgerungen hängen damit in der Luft.

3.1 Algorithmen

Eine „strukturelle Definition“ von ‹Algorithmus› gibt es nicht. Die einzige von Waldmann in Bezug genommene Quelle, der er diese Information als Konsens in der Informatik zu entnehmen glaubt und die zugleich, zu seiner Normvorgabe passend, zu Barad führt, ist *Burke*, der die intellektuelle Bewegung der *strukturierten Programmierung* überbetont (Burke 2019, S. 2), deren Konzept er irrtümlich “more-or-less” als Vorläufer anderer Programmierkonzepte wie der objektorientierten und der funktionalen Programmierung darstellt (ebd., S. 3) sowie das Dogma der Trennung von Anweisungen (“code”) und Daten ebenso als wirklichkeitsbildenden agentiellen Schnitt (ebd., S. 3–4), wie die Zusammenfassung von Algorithmen in Programmbibliotheken (ebd., S. 8). Das magische Denken von “a disempowered user outside an opaque decision-making system” (ebd., S. 9) aus der ein alternativer anthropomorphisierter, “imprecise” Algorithmus-Begriff entwickelt wird (ebd., S. 10), bildet die Grundlage für nicht-technische Diskurse über die kulturelle Wirkung von Algorithmen (ebd., S. 10–12): die „systemische Definition“ (Waldmann 2024, S. 3). In bestimmten (Anm. J.H.: etwa didaktischen) Fällen kann diese Vereinfachung selbst im technischen Diskurs informell brauchbar sein (Burke 2019, S. 10).

Die Opazität von Algorithmen und ihrer Wirkung klärt jedoch in aller Regel in dem Maße auf, in dem die Bereitschaft zunimmt, sich mit ihnen *auch* präziser als technische Artefakte auseinanderzusetzen. In Bezug auf Programmierung gibt es im Grunde drei Repräsentationsformen von Algorithmen. Sie können mit alltagssprachlichen Mitteln formuliert sein, etwa wie in Abbildung 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. ■

Abbildung 1: Allgemein formulierter Algorithmus (Knuth 1998, S. 142)

Diese Darstellung lässt sich dann wie in Abbildung 2 als zweite Repräsentationsform mehr oder minder wörtlich in Quelltext übersetzen, der in Maschinensprache überführt ausführbarer Code wird.

Als dritte Repräsentationsform kann die Transposition in eine der Struktur dieser Sprache entsprechende Fassung betrachtet werden – wie in Abbildung 3.

Je nach Abstraktionsmöglichkeiten kann der Algorithmus in der äquivalenten dritten Repräsentation vollkommen losgelöst erscheinen (Abbildung 4).

Was in dieser Programmiersprache als kompakte Schleife formuliert wird, übersetzt die Automatik im Hintergrund für den Computer in Sprunganweisungen, die der ursprünglichen Formulierung des Algorithmus entsprechen. Die Struktur dieses Algorithmus fügt sich gut in die internen Funktionsweisen von Computern und endet mit einem Ergebnis (wie beabsichtigt von Knuth 1998, S. v).

```

(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))))

```

Abbildung 2: Wörtlich übersetzter, ausführbarer Algorithmus (Seibel 2011b, S. 251)

```

(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)))

```

Abbildung 3: Idiomatisch transponierter ausführbarer Algorithmus (Rosetta-Kollektiv 2024)

```
(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)))
```

Abbildung 4: Maximal vereinfachter Algorithmus (Seibel 2011b, S. 252)

In einer prozeduralen *maschinennahen* Programmiersprache (etwa: Knuth 2005, S. iv) kann die Übersetzung des Algorithmus aussehen wie in Abbildung 5:

In der zuletzt definierten Prozedur **main** wird die Unterprozedur **test** aufgerufen, die zuvor definierte Prozeduren aufruft, die wiederum zuvor definierte Prozeduren aufrufen. Die Programmierenden haben hier wesentlich weniger Automatismen zur Verfügung, so dass zu Beginn auch eine für die Lösung geeignete Datenstruktur benutzerdefiniert wird. Im Kleinen wird die hierarchische und sequentielle Struktur von solch sehr konkreten Sprachen deutlich und ermessbar, wie komplex hier weitreichendere Problemlösungen werden können. Für diese Sprachen ist das Konzept der strukturierten Programmierung gedacht – gerade *ohne* Sprünge (Dijkstra 1970, 1968; Kritik an der Generalisierung: Harvey 1997, S. 233–238).

Eine Sprache wie die in Abbildung 2 bis 4 demonstrierte verbirgt die technischen Besonderheiten von Computern und bietet dadurch andere Möglichkeiten, Komplexität zu begegnen, die dem „Top-down-Programmierprinzip“ (Waldmann 2024, S. 2) das gleichwertige „Bottom-up design“ in der Informatik hinzugesellen (Graham 1993, S. v–vi). Sie und ihre Verwandten kommen auch auf syntaktischer Ebene ohne die dogmatische Trennung von Daten und Code aus. Daten und Code sind dort strukturgleich, was weitere Abstraktionswerkzeuge zur Reduktion von Komplexität ermöglicht (zum Merkmal Mooers und Deutsch 1965, S. 232).

Die funktionale Programmierung ist in Sprachen wie der demonstrierten ein auf der mathematischen Funktionsverkettung basierender Programmierstil, um *rekursive* Prozesse auszunutzen und die Zuverlässigkeit komplexer Programmsysteme insbesondere bei parallel ablaufenden Prozessen zu erhöhen (Turner 2013; Bauer 2004, S. 247–248), was der kybernetischen Betrachtung besonders zugänglich ist (Foerster 2015, S. 73). Rekursive Vorgänge werden hingegen in den weniger automatisierten maschinennahen Sprachen möglichst vermieden (Bauer 2004, S. 240). Rekursives Vorgehen ist aber nicht unstrukturiert oder regellos (so scheinbar Waldmann 2024, S. 3). Es geht um verallgemeinerte Schleifenkonstruktionen, die Regeln zum Durcharbeiten von baumartig verästelten, also gleichermaßen rekursiven Datenstrukturen für jedes einzelne Element wiederholen (Domkin 2021, S. 159; zum Spezialfall der endständigen Rekursion/Iteration G. Teschl und S. Teschl 2013, S. 221; Abelson, G. J. Sussman und J. Sussman 1996, S. 45–47).


```

#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;
}

```

Abbildung 5: In maschinennahe Programmiersprache übersetzter Algorithmus (Rosetta-Kollektiv 2024)

Die objektorientierte Programmierung gibt es nicht. Was den unterschiedlichen Modellen jedoch gemein ist, dass benutzerdefinierte Datenstrukturen auf dynamischere Weise mit Code assoziiert werden, als in der strukturierten Programmierung (einführend Seibel 2011a, S. 189–191). Die Modelle wurden entwickelt, um die Beschränkungen der strukturierten Programmierung bei noch höherer Komplexität *zu überwinden*.

All diese Methoden zur Organisation von Programmquelltexten unterstützen unterschiedliche Problemlösungsstrategien, in denen Algorithmen unterschiedlich repräsentiert werden. Was Algorithmen dennoch bleiben ist, was sie schon waren, bevor es digitale Computer gab (Cormen u. a. 2009, S. xv): Ein schrittweises Vorgehen zur Lösung eines genau beschriebenen, rechnerischen Problems, das bei korrektem Verfahren mit einem korrekten Ergebnis beendet wird (ebd., S. 5–6) – was schon auf antike Methoden zutrifft (vgl. etwa G. Teschl und S. Teschl 2013, S. 94–95, 177).

Um sie verlässlich nutzen zu können, muss nur ihr Verhalten bekannt sein. Die schriftliche Division etwa verlangt kein Zahlenverständnis (Padberg und Benz 2021, S. 226–227). Prozeduren werden *stets* als “*black boxes*” (Ashby 1957, S. 86) verwendet, die als *determinierte Maschinen* (ebd., S. 24) *verlässliche* Transformationen von passenden Eingabewerten zu Ausgabewerten zeigen (ebd., S. 11–16, 86–87). Eine gute Verhaltensbeschreibung gibt Programmierenden auch Hinweise auf die Nebenwirkungen, im Übrigen aber sind die internen Prozesse in der Regel ohne Belang, solange nicht Korrektheit oder Effizienz auf dem Prüfstand stehen.

Die von Waldmann über Burke zusammengetragenen Informationen zur Beschaffenheit von Algorithmen sind insoweit nicht haltbar. Dennoch verwirft er die technische Perspektive als trivial; ihre Kenntnis führe zu „vorurteilsbehafteten Resultaten“ und zu „kulturellen Phantasmen“ (Waldmann 2024, S. 3, 5).

3.2 Bewusstmachung von Entscheidungsmöglichkeiten, die technische Artefakte betreffen

Komplexe Softwarelösungen sind vielfach gekoppelte Systeme (Ashby 1957, S. 48–53), in denen wiederholte Transformationen stattfinden (ebd., S. 16–23), zum maschinellen Lernen möglicherweise Markov-Verkettungen realisiert sind (ebd., S. 165–173), andere, an die Bayessche Wahrscheinlichkeitsstatistik angelehnte Verfahren zur Anwendung kommen (Downey 2012, S. 3–6; Russell und Norvig 2010, S. 495–499) und zahlreiche weitere Methoden – aber keine „Wahrheit“ (Waldmann 2024, S. 20). Zwei Dinge werden daran offensichtlich: (1) Die Prinzipien, auf denen Anwendungen maschinellen Lernens basieren, sind erklärbar. (2) Komplexe Verkettungen mit vielfältigen Transformationen sind nicht im Detail vom Übergang der jeweiligen Eingabewerte von einer zur nächsten Komponente aus beschreibbar. Dass die Eingabe eines selbstlernenden Systems von diesem in eine Ausgabe transformiert wird bleibt aber ebenso unbestreitbar, wie dass diese

Transformation auf rationeller Grundlage, nämlich in aller Regel statistischen Verfahren erfolgt (so auch Burke 2019, S. 8–9).

Am repräsentativen Beispiel einer Studie über den Software-Einsatz zur Ausbeutung von prekär in Scheinselbständigkeit beschäftigten Lieferdienstangestellten durch die Akteure in einem idealtypischen arbeitsrechtsfeindlichen Start-up-Unternehmen (Waldmann 2024, S. 9) wird von Waldmann der Versuch unternommen, das vorgestellte post-humanistisch-neomaterialistische Analysewerkzeug zu demonstrieren und die zwangsläufige Undurchschaubarkeit von Algorithmen in Anwendungen künstlicher Intelligenz aufzuzeigen. Dazu wird einerseits ein „*Maschinenhabitus*“ angenommen, der die habituellen Eigenschaften der an der Entwicklung und den automatisierten Lernprozessen beteiligten Menschen repräsentiert, was als \langle primäre \rangle und \langle sekundäre \rangle „Maschinensozialisation“ bezeichnet wird, aus denen Pfadabhängigkeiten hervorgehen, die letztlich einen, wenn auch schwer vorhersagbaren, „praktischen Urteilssinn“ der Maschinen „[ausbilde]“ (ebd., S. 5–6, m.w.N.). Die soziologische Deutung des fiktiv-repräsentativen Fallbeispiels soll sodann zeigen, dass originär die eingesetzte Software aus ihrem Habitus heraus, „ein techno-soziales Feld [konstituiere]“ (ebd., S. 10), dessen Gesetzmäßigkeiten für alle beteiligten Menschen undurchschaubar sei aber Asymmetrien auf Ebene der habituellen Beteiligung aller Akteure verwirkliche, zu denen dann auch die Maschine gezählt wird (ebd., S. 11). Andererseits soll die neomaterialistische Deutung im Kontrast dazu, aber auch ergänzend zeigen, dass Maschinen Wirklichkeit schaffen durch „Algorithmen als agentielle Schnitte“ und dadurch ebenfalls eigenständige Beteiligte sind am mit der Materie verwobenen Diskurs über Ungleichheitsverhältnisse (ebd., S. 13–14), was die Opazität der Algorithmen bewahrt.

Dennoch wird in diesem Zusammenhang das Beispiel einer zutreffenden “Output”-Hypothese durch Bewusstmachung der das technische Artefakt betreffenden Handlungsmöglichkeiten erwähnt, wo Kurierfahrer:innen systematisch und erfolgreich den “Output” der Software ihres Auftraggebers beeinflussten (ebd., S. 18).

Ein anderes Beispiel der methodischen Verhaltensanalyse bieten zwei Informatiker:innen, aus der in Waldmanns Quellen überraschender Weise unterrepräsentierten Gruppe von „BIPoC“ (ebd., S. 4), mit Forschungsschwerpunkt auf “Bias” und «künstliche Intelligenz» (vgl. etwa Buolamwini 2016, 2017), die die Kompetenz besitzen, die Transformationen *Input*→*Output* und damit das *reproduzierbare* Verhalten von Anwendungen komplexer statistischer Verfahren des *maschinellen Lernens* angemessen zu beobachten, beschreiben und zu analysieren. Buolamwini und Gebru haben – aus der Notwendigkeit heraus, dass in den USA und anderen Staaten automatische Gesichtserkennung unter anderem für die Strafverfolgung eingesetzt wird – *mit einer geeignet aufbereiteten Datenquelle* nachvollziehbar und reproduzierbar die Ungenauigkeiten der Gesichtserkennung dreier Softwareprodukte herausgearbeitet, *eindeutig* aufgezeigt, dass die aktuell kodierten statistischen Wertungen als unerwünschte Nebenwirkung *durchschaubar* eben *nicht*

zu korrekten Ergebnissen führen und geben Handlungsempfehlungen zur Produktverbesserung (Buolamwini und Gebru 2018, S. 12). Tatsächlich liegt im Allgemeinen der Schwerpunkt bei der Programmierung wie bei der Analyse stärker auf den verwendeten und erzeugten *Daten* und ihrer Struktur als auf den Algorithmen, was aber keine Trennung im Sinne der strukturierten Programmierung meint (anders Burke 2019, S. 8–9) sondern Komplexität statisch in die transparenten Daten verlegen lässt (vgl. Domkin 2021, S. 29–30). Selbst die verborgenen Wertungen sind Daten, deren Tendenzen durch die Fehlerrate messbar werden.

Die Analyse von Buolamwini/Gebru lässt auch Deutungen zu: Es mag etwas über den Anteil „weisse[r]* Cis-Männer mit akademischem Hintergrund“ aussagen bei IBM, wenn in Bezug auf die Gesichtserkennung hellhäutiger Frauen die Fehlerquote ihres Produkts bei 0,0 % liegt (Buolamwini und Gebru 2018, S. 10). Vor allen Dingen wird aber Waldmanns Paradigma entgegengesetzt: Die technische Analyse ist *insbesondere* geeignet, algorithmisch erzeugte Ungleichheit transparent zu machen und wirksame Handlungsempfehlungen zur Fehlerbehebung zu liefern – und dadurch *relevant* (aktuell etwa Sabbatini und Calegari 2024; älterer Beitrag zu weiteren ethischen Problemen, aus eher humanistischer Perspektive: Van der Loos 2014).

Im Folgenden wird eine Implikation von Waldmanns Position vertieft und warum sie in der medienpädagogischen Bildung zumindest der Grundschule als hochproblematisch betrachtet werden kann.

4 Verantwortlichkeit

Die Vermenschlichung von Algorithmen folgt, wie bei Burke gesehen, aus Ohnmachtsgefühlen und fehlender Bereitschaft, sich Handlungsmacht zu erarbeiten – aufbauend auf einer „Argumentation aus Ignoranz“ (Walton, Reed und Macagno 2013, S. 327): «Wenn diese Algorithmen nachvollziehbar wären, könnte ich sie nachvollziehen. Ich kann es aber nicht.» bzw. einem Fehlschluss der Verallgemeinerung: «Ich kann es nicht verstehen, weil es niemand verstehen kann.»

Das mag etwas mit *Bewegung* zu tun haben: Im alten *Common Law* Englands lebte bis in das 19. Jahrhundert hinein die antike Rechtsfigur fort, wonach ein Ding „*deodand*“ werden konnte (Holmes 1991, S. 7–10). «Ding» meint hier alles das, was von «vollwertigen» Menschen als «unselbständiges» benutzt wird: Sowohl Werkzeuge, Transportmittel, Nutztiere als einst auch Sklaven. Wurden Gegenstände durch Bewegung «belebt» oder gingen Tiere durch und kam es dadurch zu erheblichen Schäden, so waren sie in bestimmten Fällen «verwirkt» und fielen zum Ausgleich für das geschehene Unglück «an Gott» (*deo-dandum*) bzw. an die Krone oder die Allgemeinheit oder auch die Geschädigten; es ging um *verschuldensunabhängige* Haftung bei nicht unmittelbar zurechenbarem Schaden (etwa bei Waldmann 2024, S. 7 unten), die dennoch mit dem Geschehen eng

assoziierte *Menschen* in klar umrissenem Umfang treffen sollte (Holmes 1991, S. 24–26). Die Eigentümer:innen waren nicht unmittelbar zum Schadenersatz verpflichtet und sie traf keine körperliche Strafe und Ächtung, aber sie verloren diesen Teil ihres Eigentums. Strafe und Ächtung konnten stattdessen das verwirkte Ding treffen, auch in Form seiner Verwertung.

Die hergebrachte Strategie zur Lösung des Problems erschien zunehmend absurd (House of Commons 1846), bei der etwa fingiert wird, dass ein Segelschiff dann (und nur dann) ein Eigenleben beginnt, wenn es sich ohne weiteres Zutun, also wie von selbst, aus seiner Vertäuerung löst, von der Flußströmung in Bewegung versetzt wird und einen Anleger beschädigt, wobei ein Mensch zu Schaden kommt (Holmes 1991, S. 22–23, 26–27). Es gibt rechtlich geschicktere Lösungen als der Umweg über die Anthropomorphisierung eines «Sündenbocks». Im Haftungsrecht, einem Grundproblem der Menschheit, lässt sich so eine Rationalisierung nachzeichnen. Mit einem «habituell und machtvoll, intra-aktiv Wirklichkeit ausschneidenden Apparat» wird wieder der umgekehrte Weg zur Vorstellung vom Ding beschritten, das durch (rekursive Schleifen-)Bewegung lebendig wird (dazu auch Turkle 2010, S. 8–10). Symptomatisch könnte wieder ein Arbeitssklave, «Roboti» (*Karel Čapek*), “deodand” werden. Ein Konstrukt vom „Maschinenhabitus“ oder einer generellen, posthumanistisch-neomaterialistisch fundierten Skepsis gegenüber dem “Output” führt aber nicht zur Ursache, dem *Ermöglichen* „algorithmischer Autorität“ (Waldmann 2024, S. 11; mit klarer Verantwortlichkeit bei Menschen: Bryson 2010).

Pädagogisch vielversprechender erscheint mir, wenn einerseits Daten sowie ihre algorithmische Verarbeitung und Erzeugung erfahrbar und beschreibbar gemacht werden, um so die Transformation **Input**→**Output** auch kritisch nachvollziehen zu können. Und wenn andererseits das Bewusstsein für die *Eigenverantwortung* daran geweckt wird, welche Werkzeuge zur Entscheidungshilfe (“computer aided design/planning/decision-making”) auf welche Weise wirksam werden, auch inwieweit individuell zugelassen wird, sich den mit ihnen vollzogenen Kommunikationsprozessen zu unterwerfen bzw. sie zum eigenen Vorteil und Schaden anderer auszunutzen.

5 Schlussfolgerung und Ausblick

Zwei spannende und interessante Konzepte, die Konstruktion eines Maschinenhabitus und agentieller Schnitte, werden (aus-)genutzt, um zu versuchen eine stumpfe Vorstellung von Algorithmen zu schärfen, ohne dafür technischen Sachverstand zuzulassen, der die eigenen normativen Vorbehalte dämpfen und bevorzugte Vorstellungen sowie die darauf aufbauende Argumentation als nicht tragfähig erweisen könnte.

Damit wird eine „irreduzible Opazität“ zurechtgelegt, die nicht Algorithmen im Zusammenhang mit maschinellern Lernen notwendig inhärent ist, sondern durch das extern bleibende Sprechen über *die Wirkung* der Technologie bei unterlassener Ausein-

andersetzung mit ihrer Beschaffenheit erzeugt wird. Es ist dies die Reproduktion und Amplifikation der selbstbeschränkenden Ohnmacht, aus welcher der Alltagssprachliche Algorithmus-Begriff entstanden ist.

Die Möglichkeiten der kybernetischen Betrachtung wurde als alternativer Zugang angedeutet, um zur Beschäftigung mit den Ursprüngen der etablierten Begriffe anzuregen. Mit ihnen wird die technische Seite in Worte gefasst, die in sozialwissenschaftlichen Disziplinen verständnisvoll aufgegriffen werden können – ohne eine fehleranfällige Übersetzung in und Rückübersetzung aus weniger etablierten Terminologien – was einen transdisziplinären Dialog ermöglicht. Der Weg zu diesem Dialog kann mit den inhaltsreichen kybernetischen Begriffen bereits, angemessen didaktisch reduziert, in der Grundschulbildung bereitet und mit der politischen Bildung zum Erkennen und Gestalten von Machtverhältnissen verknüpft werden.

Literatur

- Abelson, Harold, Gerald Jay Sussman und Julie Sussman (1996). *Structure and Interpretation of Computer Programs*. 2. Aufl. Cambridge, MA, USA: MIT Press. URL: <https://web.mit.edu/6.001/6.037/sicp.pdf> (besucht am 21.01.2017).
- Anderson, Marc M. (2024). „AI as Philosophical Ideology: A Critical Look Back at John McCarthy’s Program“. In: *Philosophy & Technology* 37.44. URL: <https://doi.org/10.1007/s13347-024-00731-1> (besucht am 14.07.2024).
- Ashby, W. Ross (1957). *An Introduction to Cybernetics*. 2. Aufl. 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> (besucht am 14.07.2024).
- 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, S. 801–831. JSTOR: 10.1086/345321. URL: <https://www.jstor.org/stable/10.1086/345321> (besucht am 22.07.2024).
- Bauer, Friedrich L. (2004). „Die Algol-Verschwörung“. In: *Geschichten der Informatik. Visionen, Paradigmen, Leit motive*. Hrsg. von Hans Dieter Heilige. Berlin/Heidelberg, GER: Springer-Verlag, S. 237–254. URL: https://link.springer.com/chapter/10.1007/978-3-642-18631-8_10 (besucht am 03.12.2023).
- Bryson, Joanna (2010). „Robots Should Be Slaves“. In: *Close Engagements with Artificial Companions: Key Social, Psychological, Ethical and Design Issues*. Hrsg. von Yorick Wilks. Natural Language Processing. Amsterdam, NED: John Benjamins Publishing Company, S. 63–74. DOI: 10.1075/nlp.8.11bry. URL: https://www.researchgate.net/publication/250333956_Robots_Should_Be_Slaves (besucht am 26.07.2024).

- Buolamwini, Joy (2016). *Unmasking Bias*. URL: <https://medium.com/mit-media-lab/the-algorithmic-justice-league-3cc4131c5148#.lh6ftfaoa> (besucht am 08.09.2023).
- (2017). „Gender Shades: Intersectional Phenotypic and Demographic Evaluation of Face Datasets and Gender Classifiers“. Diss. Cambridge, MA, USA: Massachusetts Institute of Technology. URL: <https://dspace.mit.edu/bitstream/handle/1721.1/114068/1026503582-MIT.pdf> (besucht am 08.09.2023).
- Buolamwini, Joy und Timnit Gebru (2018). „Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification“. In: *Proceedings of Machine Learning Research*. Hrsg. von Sorelle A. Friedler und Christo Wilson. Bd. 81. New York, USA: PMLR, S. 1–15. URL: <http://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf> (besucht am 24.07.2024).
- Burke, Adam (2019). „Occluded Algorithms“. In: *Big Data & Society* July–December, S. 1–15. URL: <https://doi.org/10.1177/2053951719858743> (besucht am 20.07.2024).
- Cormen u. a. (2009). *Introduction to Algorithms*. 3. Aufl. 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> (besucht am 30.05.2024).
- Dijkstra, Edgar W. (1968). „Go To Statement Considered Harmful“. In: *Communications of the ACM* 11.3, S. 147–148. URL: <https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf> (besucht am 15.07.2024).
- (1970). *Notes on Structured Programming*. Techn. Ber. T.H.-Report 70-WSK-03. Eindhoven, NED: Technische Hogeschool Eindhoven, S. 147–148. URL: <https://www.cs.utexas.edu/~EWD/ewd02xx/EWD249.PDF> (besucht am 15.07.2024).
- 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> (besucht am 09.03.2024).
- Everth, Thomas und 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> (besucht am 23.07.2024).
- Foerster, Heinz von (2002). „Thoughts and Notes on Cognition“. In: *Understanding Understanding: Essays on Cybernetics and Cognition*. Hrsg. von Heinz von Foerster. New York, USA: Springer-Verlag, S. 169–189. URL: <https://www.alice.id.tue.nl/references/foerster-2003.pdf> (besucht am 20.07.2024).

- Foerster, Heinz von (2015). „Kybernetik“. In: *Wissen und Gewissen. Versuch einer Brücke*. Hrsg. von Siegfried J. Schmidt. 9. Aufl. Frankfurt am Main, GER: Suhrkamp Verlag, S. 72–76.
- Graf, Erich Otto (2010). „Strukturalismus“. In: *Wissenschaftstheorie*. Hrsg. von Detlef Horster und Wolfgang Jantzen. Bd. 1. Behinderung, Bildung, Partizipation Enzyklopädisches Handbuch der Behindertenpädagogik. Stuttgart, GER: Kohlhammer, S. 211–214.
- Graham, Paul (1993). *On Lisp*. Hoboken, New Jersey, USA: Prentice Hall. URL: <https://paulgraham.com/onlisptext.html> (besucht am 02.11.2017).
- Harvey, Brian (1997). *Computer Science Logo Style: Volume 1. Symbolic Computing*. 2. Aufl. Cambridge, MA, USA: MIT Press. URL: <http://people.eecs.berkeley.edu/~bh/pdf/v1ch13.pdf> (besucht am 18.08.2015).
- Hegel, Georg Wilhelm Friedrich (2014). *Phänomenologie des Geistes*. 13. Aufl. Bd. 3. Werke. Frankfurt am Main, GER: Suhrkamp Verlag.
- Holmes, Oliver Wendell jr. (1991). *The Common Law*. Bd. Nachdruck der 1. Auflage (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> (besucht am 21.07.2024).
- Knuth, Donald E. (1998). *Seminumerical Algorithms*. 3. Aufl. Bd. 2. The Art of Computer Programming. Reading, MA, USA et. al.: Addison-Wesley. URL: <https://archive.org/details/artofcomputerpro0000knut/mode/2up> (besucht am 03.07.2024).
- (2005). *MMIX: A RISC Computer for the New Millennium*. Bd. 1. Fascicle 1. The Art of Computer Programming. Upper Saddle River, New Jersey, USA: Addison-Wesley.
- Luhmann, Niklas (2013). *Das Recht der Gesellschaft*. 6. Aufl. Frankfurt a.M., Deutschland: Suhrkamp Verlag.
- Mooers, Calvin N. und 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, S. 229–246. URL: <https://doi.org/10.1145/800197.806048> (besucht am 24.07.2024).
- Padberg, Friedhelm und Christiane Benz (2021). *Didaktik der Arithmetik. fundiert, vielseitig, praxisnah*. 5. Aufl. Berlin, Deutschland: Springer Spektrum.
- Primas, Hans (1994). „Realism and Quantum Mechanics“. In: *Logic, Methodology and Philosophy of Science*. Hrsg. von Dag Prawitz, Brian Skyrms und Dag Westerståhl. Amsterdam, NED: Elsevier Science, S. 609–631. URL: <https://philsci-archive.pitt.edu/951/1/Realism%26QuantumMechanics.pdf> (besucht am 23.07.2024).
- Rosetta-Kollektiv (2024). *Knuth's Algorithm S*. URL: https://rosettacode.org/wiki/Knuth%27s_algorithm_S (besucht am 24.07.2024).

- Russell, Stuart J. und Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. 3. Aufl. Upper Saddle River, New Jersey, USA: Pearson. URL: https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf (besucht am 20.12.2019).
- Sabbatini, Federico und Roberta Calegari (2024). „On the Evaluation of the Symbolic Knowledge Extracted from Black Boxes“. In: *AI and Ethics* 4, S. 65–74. URL: <https://doi.org/10.1007/s43681-023-00406-1> (besucht am 27.07.2024).
- Seibel, Peter (2011a). „Object Reorientation: Generic Functions“. In: *Practical Common Lisp*. New York, USA: Apress. URL: <https://gigamonkeys.com/book/object-reorientation-generic-functions> (besucht am 11.02.2019).
- (2011b). „The Special Operators“. In: *Practical Common Lisp*. New York, USA: Apress. URL: <https://gigamonkeys.com/book/the-special-operators> (besucht am 11.02.2019).
- 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> (besucht am 24.07.2024).
- Teschl, Gerald und Susann Teschl (2013). *Mathematik für Informatiker. Band 1. Diskrete Mathematik und Lineare Algebra*. 4. Aufl. 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*. Hrsg. von Yorick Wilks. Natural Language Processing. Amsterdam, NED: John Benjamins Publishing Company, S. 3–10. DOI: 10.1075/nlp.8.11bry. URL: http://web.mit.edu/people/sturkle/pdfsforstwebpage/Turkle_In%20Good%20Company.pdf (besucht am 30.07.2024).
- 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*. Hrsg. von Hans Wolfgang Loidl und Ricardo Peña. Heidelberg, GER/New York, USA/Dordrecht, NED/London, GBR: Springer, S. 1–21. URL: https://link.springer.com/chapter/10.1007/978-3-642-40447-4_1 (besucht am 10.03.2024).
- 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, S. 5–6. URL: <https://doi.org/10.1007/s11948-014-9524-1> (besucht am 28.07.2024).
- 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, S. 1–23. URL: <https://doi.org/10.21240/mpaed/61/2024.06.10.X> (besucht am 20.07.2024).

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