cmd Ein Kommando zur Berechnung auf der Hinterbühne

Andreas Breiter

Die Abkürzung **cmd (command)** *verweist auf ein zentrales Konzept der Informatik, mit dessen Hilfe Befehle ausgeführt werden können, die unterhalb der Benutzungsoberfläche des Computers liegen. Anhand des fiktiven Kommandos* **cmd** *traceuserbehavior werden im Beitrag die verborgenen Aspekte und Tragweiten algorithmischer Datenverarbeitung in den Blick genommen und speziell im Forschungskontext kritisch diskutiert.*

Die Informatik, aus der die hier betrachtete Abkürzung **cmd** hervorgeht, versteht sich selbst als Leitdisziplin der Informationsgesellschaft und damit als *Gestaltungswissenschaft* der digitalen Transformation. Ursprünglich ist die deutschsprachige Disziplin in den 1960er Jahren aus zwei Strömungen heraus entstanden: zum einen aus der Computerwissenschaft (*Computer Science*) im anglo-amerikanischen Raum mit einem Schwerpunkt auf Berechenbarkeit und Rechenmaschinen sowie zum zweiten aus der französischen *Informatique* – von der sie auch ihren Namen erhielt – mit Fokus auf der Informationsverarbeitung. Heutzutage ist eine diesbezügliche Unterscheidung nicht mehr zu erkennen, stattdessen differenziert sich die akademische Disziplin *Informatik* (oder eben englisch: *Computer Science*) immer stärker aus: in Hardware-nahe Bereiche (wie *Cyber-physical Systems*), in Software Engineering sowie in Human-Computer Interaction (HCI) an der Schnittstelle zwischen Informatik, Psychologie und Sozialwissenschaften. Im Mittelpunkt steht die Verarbeitung und Speicherung von Daten mit Hilfe von Computersystemen und deren Programmierung sowie die Gestaltung der Schnittstellen von Mensch und Maschine (*Human-in-the-loop*).

In diesem Kontext spielt die Abkürzung **cmd** eine zentrale Rolle: Mit Einsatz der Taste **cmd** lässt sich die Kommandozeile des Computers öffnen, mit deren Hilfe Befehle ausgeführt werden können, die nicht unmittelbar auf der Ebene der Anwendungs- oder Dienstprogramme liegen. Hinter einem *Kommando* verbirgt sich außerhalb der Begriffsverwendung in militärischen Kontexten eine Zeichenfolge, mit der entweder das Betriebssystem oder ein Anwendungsprogramm veranlasst wird, eine oder mehrere Funktionen auszuführen. Oft werden die Begriffe *Befehl* oder *Anweisung* synonym hierzu verwendet. Die mit **cmd** verfolgte Anweisung wird am Computer durch den *Prozessor* ausgeführt. Dieser ist ein Microchip, der im Kern nur binäre Codes verarbeiten kann (also Nullen und Einsen). Der Prozessor holt sich die zu verrechnenden Daten aus dem Speicher (auch ein Microchip) und legt sie nach erfolgter Rechenleistung dort wieder ab. Um den Aufwand der Programmierung zu reduzieren, wurden formale Programmiersprachen (etwa Assembler, Pascal, C/C++ oder Java) entwickelt, die mit einfachen (natürlich-sprachigen Befehlen) komplexe Instruktionen ausführen können. Mit fortschreitender Entwicklung kann das *Kommando* – etwa bei grafikorientierten Benutzungsoberflächen – auch durch das Anklicken beziehungsweise Berühren eines Menüpunktes oder Icons erfolgen.

Sowohl durch die direkte Eingabe eines Kommandos als auch durch das Klicken auf ein Icon greifen wir mit **cmd** hinter der Oberfläche (*User Interface*) eines Computers auf eine darunterliegende, Nutzer*innen meistens unbekannte, Systemschicht zu. Dabei ist die Unterscheidung von *Interfaces* und *Systemschicht* für den vorliegenden Beitrag besonders interessant: Denn bereits in die Systemschicht sind spezifische Nutzungsmöglichkeiten als Handlungsoptionen für Nutzende eingeschrieben, die wesentlich dadurch bestimmt sind, wie sie von Informatiker*innen als Produzent*innen verstanden werden (siehe **.exe,** Verständig 2023). Interface und Systemschicht lassen sich zudem gesellschaftstheoretisch interpretieren, wenn Informatik nicht bloß als anwendungsorientiertes Mittel zur Herstellung von Soft- oder Hardware oder Gestaltung von Mensch-Maschine-Schnittstellen verstanden wird, sondern infolge deren Produktion und Anwendung gesellschaftliche Reproduktionsmechanismen kritisch beleuchtet werden: In Anlehnung an die Goffmansche Theorie (1959) ließe sich beispielsweise auch hier zwischen Vorderbühne und Hinterbühne unterscheiden (siehe auch **Hg.**, Krebber 2023). Goffmans Konzept basiert auf der Grundannahme, dass jede zwischenmenschliche Interaktion dadurch geprägt ist, dass die Teilnehmenden mittels verschiedener Darstellungsformen ein bestimmtes Bild von sich präsentieren. Analog zum Theater wird auf der Vorderbühne das idealistische Selbstbild gespielt, während auf der Hinterbühne Tatsachen oder informelle Handlungen verborgen liegen. Beide Bühnen grenzen aneinander und sind zugleich voneinander abgeschnitten. Dieses Konzept lässt sich auf die Trennung zwischen Benutzungsoberfläche (*welche Funktionen werden angeboten?*) und Betriebssystem (*was wird ausgeführt?*) beziehungsweise Datenbanksystem (*was wird wie gespeichert?*) übertragen. Ein Interface wäre demnach die Vorderbühne, also das, was sich Nutzer*innen aneignen können. Die Systemschicht stelle demgegenüber die Hinterbühne dar, in der soziale Reproduktion stattfinden kann. Mehr noch: In gewisser Weise spielen die Entwickler*innen auch auf der Vorderbühne den Nutzer*innen etwas vor. Berry (2011) beschreibt diese in Technik eingeschriebenen, sozialen Beziehungen als „digital iceberg“: So sehen Nutzende allenfalls einzelne digitale Spuren an der Oberfläche, wenngleich sie die elektronischen Signale, die über Leiterbahnen

kommuniziert werden oder versteckte Einträge im Betriebssystem beziehunsgweise in den Registern des Prozessors nicht (mehr) wahrnehmen können. Am Beispiel eines fiktiven **cmd** *traceuserbehaviour* sollen diese unsichtbaren Aspekte des digitalen Eisbergs näher in den Blick genommen werden.

traceuserbehaviour

Stellen wir uns vor, das Betriebssystem auf der Hinterbühne hätte eine versteckte Funktion (**cmd** *traceuserbehaviour*), die es erlaubt, alle Anwendungen, die wir auf dem Computer aufrufen und alle Daten, die wir eingeben (per Tastatur, Maus, Touch, Sprache, Gesten), zu verfolgen und aufzuzeichnen. Diese Funktion gibt es so nicht in einem Betriebssystem, wäre aber relativ leicht herzustellen (zu programmieren). Hierzu ließen sich vielfältige Anwendungsbereiche finden. So gäbe es etwa Funktionen zur permanenten Überwachung durch staatliche oder private Sicherheitsorgane sowie *intelligente* Systeme, die uns anhand unserer Nutzungshistorie Hilfestellungen zur Arbeitserleichterung gäben. Die Vielfalt dieser Szenarien ist groß: von der Kontrolle bis zur Assistenz. Mit **cmd** *traceuserbehaviour* lösen wir eine Folge von Befehlen aus, deren systematische Abfolge in der Informatik als *Algorithmus* bezeichnet wird. Donald E. Knuth, ein Informatikpionier, definiert einen Algorithmus als eine Menge von Regeln, die eine Folge von Operationen definiert, sodass jede Regel effektiv und eindeutig ist und die Sequenz in endlicher Zeit beendet wird (Knuth 1979). Ich möchte hier explizit auf die Bühnenmetapher nach Goffman zurückgreifen, wonach bereits im Grundverständnis eines Algorithmus eine bestimmte Vorstellung des Nutzungsszenarios eingelagert ist. Auch nach Gillespie (2014: 167) sind Algorithmen „procedures for transforming input data into a desired output, based on specified calculations“. Er argumentiert, dass die Algorithmen träge und bedeutungslos seien, bis sie mit Datenbanken verknüpft würden, auf denen sie ihre Berechnungen durchführten. Kitchin (2017) verweist in diesem Kontext darauf, dass Algorithmen auf verschiedene Weise verstanden werden können: technisch, rechnerisch, mathematisch, politisch, kulturell, ökonomisch, kontextuell, materiell, philosophisch und ethisch. Sie seien immer eingebettet in sozio-technische Zusammenhänge, was ihre Betrachtung im Zusammenhang mit Goffmans Bühnenmetapher so fruchtbar macht.
Ich bleibe noch einen Moment beim informatischen Kern eines solchen Befehls: Was steckt hinter **cmd** *traceuserbehaviour*? Im Kern der Ausführung steht eine Übersetzung der Programmzeilen in maschinenlesbaren Code. Dies soll am Beispiel eines Sortierverfahrens (*Quick Sort*) im Folgenden veranschaulicht werden. In der Programmiersprache C würden die zugehörigen Programmzeilen so aussehen:

```
#include<stdio.h>
void  quicksort(int  number[5],int
first,int last){
   int i, j, varpivot, vartemp;

   if(first<last){
      varpivot=first;
      i=first;
      j=last;

      while(i<j){
            while(number[i]<=num-
ber[pivot]&&i<last)
         i++;
       while(number[j]>number[piv-
ot])
         j--;
        if(i<j){
         vartemp=number[i];
         number[i]=number[j];
         number[j]=temp;
        }
      }

      vartemp=number[pivot];
      number[pivot]=number[j];
      number[j]=vartemp;
      quicksort(number,first,j-1);
      quicksort(number,j+1,last);

   }
}
```

Diese Vorschrift wird von einem Übersetzer (*Compiler*) in eine maschinenlesbare Form gebracht und dann an den Prozessor weitergegeben. Ein Auszug aus einer Maschinensprache (Assembler) für die gleiche Sortierfunktion könnte so aussehen[1]:

```
_quickSort@12:
        ; set up stack, save regs
        push   ebp
        mov    ebp, esp
        push   ebx
        push   ecx
        push   edx

        ; Load function parameters into registers
        mov    ebx, [ebp + 8]       ; A stored in ebx
        mov    ecx, [ebp + 12]      ; p stored in ecx
        mov    edx, [ebp + 16]      ; r stored in edx

        ; if (p < r)
        cmp    ecx, edx
        jnl    endIf2
```

Anhand des exemplarisch dargestellten Assembler-Codes wird deutlich, dass mit der Ausführung des Befehls einzelne Anweisungen an den Prozessor übergegeben und von diesem wiederum technisch weiterverarbeitet werden. Jeder Prozessor verfügt über einen Befehlssatz mit deren Hilfe jeweils einzelne Schritte ausgeführt werden: Speicher auslesen, Operation ausführen, Schnittstellen ansteuern. Sie werden immer auf die gleiche Art und Weise ausgeführt. Der Prozessor kann sich dagegen nicht widersetzen. In unserem fiktiven Beispiel von **cmd** *traceuserbehaviour* ermöglicht die definierte Schrittfolge die Protokollierung der Tastatureingabe und Mausbewegungen. Sprachbefehle (beispielsweise über die Sprachassistenzsysteme) werden aufgezeichnet, eingehende E-Mails und Dokumente gescannt und abgespeichert sowie sämtliche Bilder und Videos archiviert und verschlagwortet. Es entsteht ein riesiger Datensatz über die eigenen *Bewegungen*, Interessen und persönlichen (nicht nur digitalen) Vorlieben. Hier kommen jetzt die sozio-technischen Zusammenhänge ins Spiel, wie sie etwa Kitchin (2017) skizziert. Von ihnen ausgehend lassen sich verschiedene Einsatzszenarien des Befehls vorstellen, die je nach Kontext von nützlich bis bedrohlich ausfallen: In sicherheitskritischen Infrastrukturen müssen alle Eingaben beispielsweise in einem Datenbanksystem gespeichert werden, um Fehlbedienungen des Systems ausfindig zu machen – dies kann der Absicherung ebenso wie der Überwachung der Nutzer*innen dienen und unterliegt gesetzlichen Vorschriften. Informationssicherheits-Managementsysteme (Pohlmann 2019) werden in diesem Kontext aufgebaut, um Menschen und Daten zu schützen. Das gleiche Überwachungssystem könnte – beispielsweise bei regimekritischen Journalist*innen – auch zu einem Eingriff in das persönliche Grundrecht und unter Umständen lebensbedrohlichen Konsequenzen führen.

Bei Forschungsarbeiten ergibt sich das gleiche Dilemma. Die Nutzung der Hinterbühne durch **cmd** *traceuserbehaviour* würde es Forschenden erlauben, auf „digitale Spuren" (Breiter/Hepp 2018) von Nutzer*innen zuzugreifen. Auf der einen Seite ergäben sich daraus nützliche und hilfreiche Einsatzszenarien, etwa für Expert*innen für Usability und Gebrauchstauglichkeit sowie Interaktionsdesign (Preece/Rogers/Sharp 2015). Softwaresysteme könnten auf Basis des Nutzungsverhaltens ständig verbessert werden und die Gestaltung barrierefreier Informationssysteme wäre einfacher, weil das System sich ständig an die individuellen Bedarfe der Nutzer*innen anzupassen vermöge. Soziale Interaktionen ließen sich leichter und ohne aktive Mitwirkung der Akteur*innen rekonstruieren (im Sinne von non-reaktiven Verfahren der empirischen Sozialforschung, siehe Döring/Bortz 2016); mittels Datenanalysen könnten dann die Kommunikationsformen in den verschiedenen Medienensembles beschrieben werden. Als weiteres nutzenbezogenes Beispiel könnte die Digitalisierung von Lern- und Lehrprozessen und der gestiegene Einsatz von Software in formalen und informellen Bildungsprozessen dienen. Lernanalysen (Ifenthaler/Drachsler 2018) auf Basis von **cmd** *traceuserbehaviour* könnten durch die Ermittlung der Bewegungen in und zwischen den softwaregestützten Lernsystemen dafür genutzt werden, individualisierte Lernwege zu empfehlen und personalisierte Lernaufgaben bereitzustellen (etwa beim Musiklernen, siehe Krieter/Breiter 2018). Dies soll dabei helfen, die Lern- und Lehrprozesse zu verbessern sowie die Lernwirksamkeit zu erhöhen (Knobbout/van der Stappen 2020; Tsai et al. 2021).

Auf der anderen Seite bedeute die Nutzung einer solchen Befehlsfolge einen drastischen Eingriff in die Privatsphäre. Die informierte Einwilligung der Betroffenen wäre zwingend erforderlich und die Akzeptanz damit eher niedrig, was wiederum den Aufwand für die Forschenden erhöhe. Auch eine mögliche Anonymisierung müsse transparent erfolgen. Hierfür gibt es Verfahren wie *differential privacy* (Dwork 2008), die aber vor allem auf großen Datenmengen funktionieren und technisch sehr komplex sind. Krieter/Viertel/Breiter (2021) konnten zeigen, dass die Bereitschaft zur Teilnahme an Forschungsprojekten mit derart intrusiven Verfahren eher gering ist und nur durch umfangreiche Sicherheitsmaßnahmen und Transparenz erhöht werden kann. Ein weiterer kritischer Aspekt der Nutzung der Hinterbühne ist die Verletzlichkeit der Computersysteme. Mit Hilfe von **cmd** *traceuserbehaviour* könnten weitreichende Eingriffe in die Grundstruktur des Rechners vorgenommen sowie Programme jeglicher Art (auch versteckt) ausgeführt werden. Das ist ein elementarer Baustein jeder Systemarchitektur, denn die Nutzer*innen wollen nicht alle Befehle manuell auf der Vorderbühne ausführen und sind dankbar für parallele Prozesse im Hintergrund. Dies gilt allerdings sowohl für Software mit gewünschten Funktionen als auch für sogenannte Schadsoftware. Auch diese bedient sich gerne der Kommandozeile. Durch die Einspeisung von Viren oder Trojanern (Pohlmann 2019) können Funktionen ausgeführt werden, von denen Benutzer*innen nichts erfahren – und so das Kommando übernehmen oder Festplatten verschlüsseln oder gar löschen.

Wenn **cmd** *traceuserbehaviour* also nicht nur dazu diene, Daten aufzuzeichnen, sondern zugleich den Aufbau einer großen Datenmenge steuern und Algorithmen bereitstellen würde, die Muster in den

Daten erkennen, wäre der Schritt zum Einsatz von Verfahren des maschinellen Lernens nicht mehr weit (Burell 2015; MacKenzie 2017). Auf der Hinterbühne von **cmd** *traceuserbehaviour* verbärgen sich dann – ganz im Sinne eines Eisbergs – undurchschaubare Rechenanweisungen, die bei künstlichen neuronalen Netzen nicht mehr rekonstruierbar sind. Im Gegensatz zu Systemen, die auf imperativer Programmierung basieren, werden Systeme des maschinellen Lernens *trainiert*, nicht programmiert. Dafür wird ein mathematisches Modell formuliert, eine Kostenfunktion aufgestellt und in Bezug auf bestimmte Eingangs- und Ausgabedaten optimiert. Die Trainingsanweisungen sind transparent, die Entstehung der *gelernten* Muster nicht. Hierdurch können zum einen Artefakte entstehen, die sich der Nachvollziehbarkeit entziehen. Zum anderen hängen die Ergebnisse immer vom vorliegenden Trainingsdatensatz ab. Das bedeutet auch, dass durch den Trainingsdatensatz bereits bestimmte Festlegungen erfolgen: Wenn dort nur Testergebnisse von männlichen Sportlern aus Südasien enthalten wären, würde eine Übertragung auf Sportlerinnen in Kamerun nicht möglich sein. Für diese algorithmisch produzierten Vorurteile gibt es bereits zahlreiche Anwendungsbeispiele (O'Neil 2016): automatische Gesichtserkennung, Bewertung der Kreditwürdigkeit, Personalauswahl, Lernleistungsprognosen oder *predictive policing* (Brantingham/Valasik/Mohler 2018). Lee und Björklund Larsen (2019: 1) beschreiben Algorithmen in Systemen des maschinellen Lernens als „biased blackboxes", die Rassismus reproduzieren und damit Ungleichheit automatisieren. Die in ihren Diskursen vorherrschende Vision einer *Datafizierung* basiert auf der Verfügbarkeit großer Datenmengen durch ständiges Nachverfolgen und Aufzeichnen digitaler Spuren einerseits sowie leistungsfähigen Algorithmen andererseits, die eine Umgebung der Überwachung generieren. Durch das weitreichende Eingreifen auf der Hinterbühne öffnet **cmd** *traceuserbehaviour* das Fenster für eine Totalüberwachung im Sinne dieser *dataveillance* (van Dijk 2014) – womit die Gefahr eines Kontrollverlusts einhergeht. Diese Phänomene wurden insbesondere im Bildungskontext ausführlich untersucht (vgl. Lupton/Williamson 2017; Yu/Couldry 2020). Hartong und Breiter (2021) etwa beschreiben die automatische Produktion und Reproduktion von Sozialindizes für die Ressourcenallokation an Schulen auf Basis der erzeugten und erhobenen Daten durch einen Algorithmus und zeigen daran die Entstehung von *inequalities of dataveillance* auf.

Konsequenzen

Der Beitrag hat versucht, anhand des fiktiven Kommandos **cmd** *traceuserbehaviour* die Mächtigkeit und versteckten Wirkungen von Algorithmen zu verdeutlichen, die sich – selbst für Expert*innen – zumeist im Verborgenen abspielen. Hierfür habe ich die Bühnenmetapher genutzt, wie sie insbesondere für zwischenmenschliche Interaktionen gebräuchlich ist, aber auch im hier vorgestellten Zusammenhang angemessen sein könnte: In Anlehnung an Goffman sollte deutlich geworden sein, dass Nutzer*innen lediglich Ausschnitte von Computersystemen sehen und erleben können. Dabei lassen sich durch Befehle dieser Art Forschungsfragen über Tragweiten von Algorithmen verfolgen, die sowohl zum Wohle der Nutzer*innen als auch zu deren Überwachung eingesetzt werden können. Die kritische Informatikforschung beschäftigt sich schon seit Jahrzehnten mit diesem Dilemma. Durch die gestiegenen Einsatzmöglichkeiten von Verfahren des maschinellen Lernens – die auf die heute massenhaft verfügbaren digitalen Daten zurückgehen – muss der Hinterbühne derartiger, Daten verarbeitender Systeme mehr Beachtung geschenkt werden. Unter dem Stichwort *Transparent AI* oder *Explainable AI* wird versucht, die möglichen Vorurteile, die bereits in den Daten vorliegen ebenso wie die durch die Algorithmen bestimmten Vorhersagen für die Nutzer*innen leichter zugänglich, also erklärbar, zu machen. Allerdings gibt es auch hier Studien, die eher skeptisch sind, was die Wirksamkeit und Akzeptanz transparenter KI-Systeme angeht (Heuer/Jarke/Breiter 2021). Nicht zuletzt deswegen ist mit der Nutzung von **cmd** eine forschungsethische Dimension verbunden, die die Disziplin der Informatik zur Reflexion ihres Handelns anregen sollte. Ein Zugriff auf die Algorithmen und Daten auf der Hinterbühne zu Forschungszwecken erfordert zugleich ein an den Nutzer*innen orientiertes und datenschutzfreundliches Design. Dafür sind bereits a priori Voraussetzungen für eine transparente, datensparsame und nutzer*innenzentrierte Gestaltung zu schaffen (*privacy by design*, siehe Langheinrich 2001). Für *Doing research* mit **cmd** ergeben sich hieraus zahlreiche Implikationen: Es braucht ein grundlegendes Verständnis der ethischen, rechtlichen und sozialen Verstrickungen, im Idealfall in Form einer interdisziplinären Zusammenarbeit. Diese muss zu einem zentralen Bestandteil der Förderung im Studium und in der Dissertationsphase werden.

Anmerkungen

1 Siehe URL: kurzelinks.de/vr3i [27.10.2021].

Referenzen

Berry, David M. (2011). *Understanding Digital Humanities*. Basingstoke: Palgrave Macmillan.

Brantingham, P. Jeffrey/Valasik, Matthew/Mohler, George O. (2018). Does Predictive Policing Lead to Biased Arrests? Results From a Randomized Controlled Trial. *Statistics and Public Policy, 5*(1), 1–6.

Breiter, Andreas/Hepp, Andreas (2018). Die Komplexität der Datafizierung. Zur Herausforderung, digitale Spuren in ihrem Kontext zu analysieren. In Christian Katzenbach/Christian Pentzold/Sigrid Kannengießer/Marian Adolf/Monika Taddicken (Hg.), *Neue Komplexitäten für Kommunikationsforschung und Medienanalyse: Analytische Zugänge und empirische Studien.* Berlin: SSOAR, 27–48.

Burrell, Jenna (2015). How the Machine 'Thinks'. Understanding Opacity in Machine Learning Algorithms. *Big Data & Society, 3*(1), 1–12.

Döring, Nicole/Bortz, Jürgen (2016). *Forschungsmethoden und Evaluation für Human- und Sozialwissenschafiler*. Berlin: Springer.

Dwork, Cynthia (2008). Differential Privacy. A Survey of Results. In Manindra Agrawal/Ding-Zhu Du/Zhenhua Duan/Angsheng Li (Hg.), *Theory and Applications of Models of Computation (TAMC)*. Berlin et al.: Springer, 1–19.

Gillespie, Tarleton (2014). The Relevance of Algorithms. In Tarleton Gillespie/Pablo J. Boczkowski/Kirsten A. Foot (Hg.), *Media Technologies. Essays on Communication, Materiality, and Society*. Cambridge: MIT Press.

Goffman, Erving (1959). *The Presentation of Self in Everyday Life*. Garden City et al.: Doubleday.

Heuer, Hendrik/Jarke, Juliane/Breiter, Andreas (2021). Machine Learning in Tutorials. Universal Applicability, Underinformed Application, and other Misconceptions. *Big Data & Society, 8*(1).

Ifenthaler, Dirk/Drachsler, Hendrik (2018). Learning Analytics. In Helmut Niegemann/Armin Weinberger (Hg.), *Lernen mit Bildungstechnologien. Praxisorientiertes Handbuch zum intelligenten Umgang mit digitalen Medien*. Berlin et al.: Springer, 1–20.

Kitchin, Rob (2017). Thinking Critically About and Researching Algorithms. *Information, Communication & Society, 20*(1), 14–29.

Knobbout, Justian/van der Stappen, Esther (2020). A Capability Model for Learning Analytics Adoption. Identifying Organizational Capabilities from Literature on Learning Analytics, Big Data Analytics, and Business Analytics. *International Journal of Learning Analytics and Artificial Intelligence for Education, 2*(1), 47–66.

Knuth, Donald E. (1997). *The Art of Computer Programming. Volume 1: Fundamental Algorithms*. Redwood City: Addison Wesley Longman.

Krieter, Philipp/Breiter, Andreas (2018). Analyzing Mobile Application Usage. Generating Log Files from Mobile Screen Recordings. Präsentiert auf der *20th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI 18)*, New York.

Krieter, Philipp/Viertel, Michael/Breiter, Andreas (2021). ‹Da habe ich es dann einfach ausgeschaltet›. Perspektiven von Lernenden auf Datensammlung mittels Langzeit-Bildschirmaufzeichnungen in non-formalen Bildungskontexten. *MedienPädagogik, 44*, 1–21.

Langheinrich, Marc (2001). Privacy by Design. Principles of Privacy-Aware Ubiquitous Systems. In Gregory D. Abowd/Barry Brumitt/Steven Shafer (Hg.), *Ubicomp 2001. Ubiquitous Computing*. Berlin: Springer, 273–291.

Lee, Francis/Björklund Larsen, Lotta (2019). How should we theorize algorithms? Five ideal types in analyzing algorithmic normativities. *Big Data & Society, 6*(2), 1–6.

Lupton, Deborah/Williamson, Ben (2017). The Datafied Child. The Dataveillance of Children and Implications for their Rights. *New Media & Society, 19*(5), 780–794.

Mackenzie, Adrian (2017). *Machine Learners. Archaeology of a Data Practice*. Cambridge: MIT Press.

O'Neil, Cathy (2016). *Weapons of Math Destruction. How Big Data Increases Inequality and Threatens Democracy*. New York: Crown.

Pohlmann, Norbert (2019). *Cyber-Sicherheit*. Berlin: Springer.

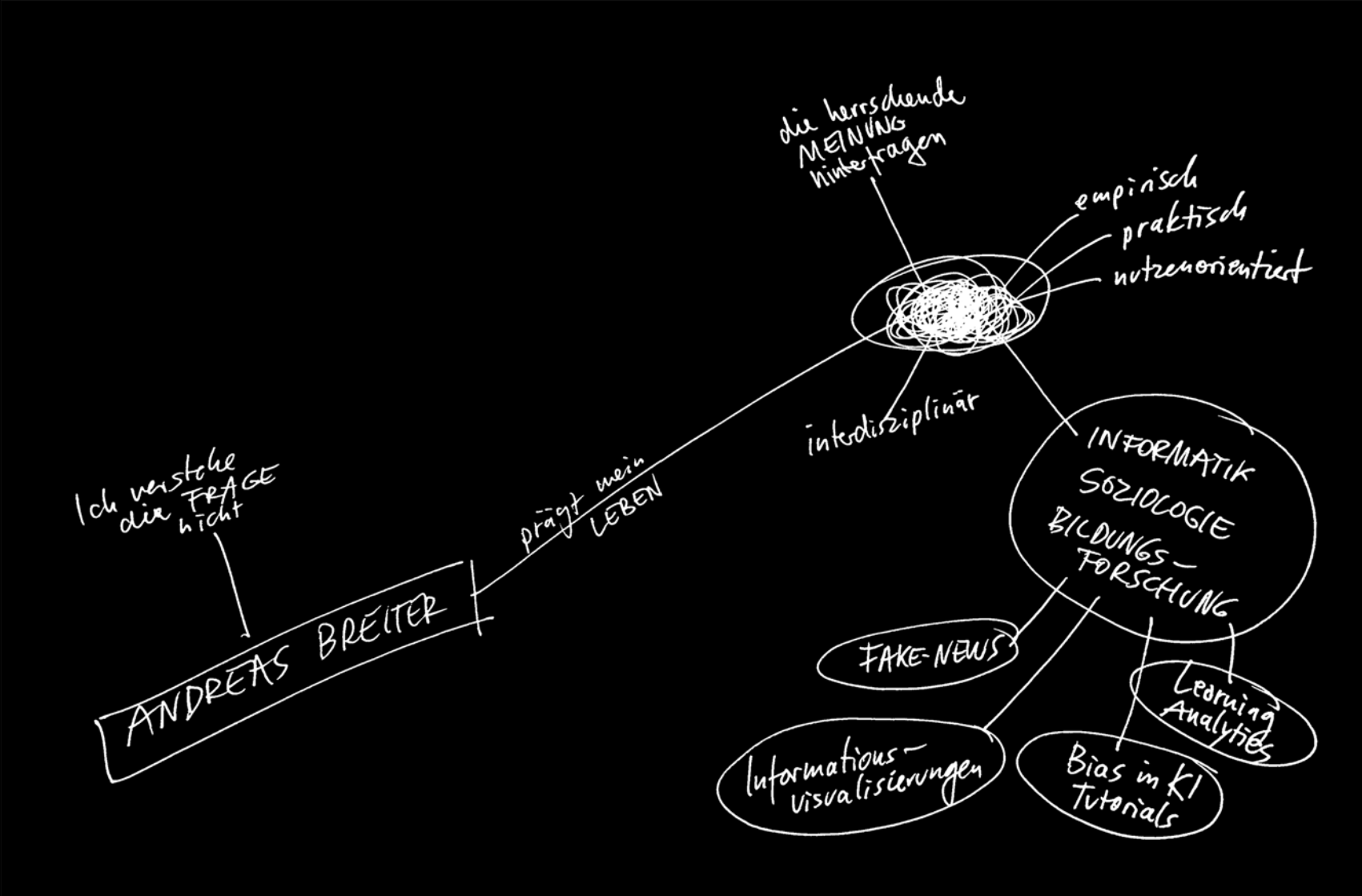
Preece, Jenny/Rogers, Yvonne/Sharp, Helen (2015). *Interaction Design. Beyond Human-Computer Interaction*. New York: Wiley.

Tsai, Yi-Shan/Rates, Diego/Moreno-Marcos, Pedro M./Muñoz-Merino, Pedro J./Jivet, Ioana/Scheffel, Maren/Drachsler, Hendrik/Delgado Kloos, Carlos, Gašević, Dragan (2020). Learning Analytics in European Higher Education. Trends and Barriers. *Computers & Education, 155*.

van Dijck, Jose (2014). Datafication, Dataism and Dataveillance. Big Data between Dcientific Paradigm and Decular Belief. *Surveillance & Society, 12*(2), 197–208.

Verständig, Dan (2023). .exe. In Sandra Hofhues/Konstanze Schütze (Hg.), *Doing Research*. Bielefeld: Transcript, 18–25.

Yu, Jun/Couldry, Nick (2020). Education as a Domain of Natural Data Extraction. Analysing Corporate Discourse about Educational Tracking. *Information, Communication & Society*, 1–18.


die herrschende MEINUNG hinterfragen
empirisch
praktisch
nutzenorientiert
interdisziplinär
INFORMATIK
SOZIOLOGIE
BILDUNGS-FORSCHUNG
Ich verstehe die FRAGE nicht
ANDREAS BREITER
prägt mein LEBEN
FAKE-NEWS
Informations-visualisierungen
Bias in KI Tutorials
Learning Analytics