50 Jahre Black Box

Erschienen im Merkur, Dezember 2017, 71. Jahrgang, Heft 823, S. 16–30, volltext.merkur-zeitschrift.de/article/mr_2017_12_0016-0030_0016_01. Der kurze Link zu dieser Version hier: bit.ly/50JahreBlackBox.

2012 schrieb ich in der Süddeutschen Zeitung über das Trendthema Algorithmenkritik[1] und erwähnte darin einen Artikel über den Netflix Prize, der vier Jahre zuvor in der New York Times erschienen war. Darin hieß es, die Empfehlungsalgorithmen für Filme würden zwar immer besser, aber auch immer unverständlicher, nicht nur für die Anwender, sondern auch für die Entwickler selbst: "Chris Volinsky admits that his team's program has become a black box, its internal logic unknowable." Ich übernahm diese Aussage in meinen Beitrag.

Danach tauchten immer mehr Artikel auf, in denen "Algorithmen" pauschal als Black Box bezeichnet wurden. Ich ärgerte mich über diese Texte, und noch mehr ärgerte ich mich darüber, dass ich mich selbst unter die Algorithmen-Mystifizierer eingereiht hatte. Bei der Bezeichnung fängt es schon an: Man könnte einfach "Software" sagen, aber das klingt eben nicht so sinister wie "Algorithmus", sondern nur ein bisschen nach angestaubten Disketten. Algorithmus bedeutet nicht "irgendwas, woran ein Computer beteiligt ist". Das Wort bezeichnete eigentlich einmal ein klar umrissenes Rezept, zum Beispiel ein Sortierverfahren oder eine Anleitung zum Zusammenbau eines IKEA-Regals. Algorithmenkritik handelt nicht von solchen kleinen Bausteinen, sondern von großen Systemen. Aber was soll man machen, in der Umgangssprache verschwindet die Software zugunsten des Algorithmus, und wer gegen den Sprachwandel protestiert, der kann auch gleich gegen Ebbe und Flut demonstrieren.

Die Algorithmenkritik der Jahre 2010 bis 2012, von der mein Artikel handelte, war eine andere als die heutige. Frank Schirrmacher war nicht einverstanden mit der Stilllegung des europäischen Flugverkehrs nach dem Eyjafjallajoküll-Ausbruch auf der Basis von Simulationen. Miriam Meckel kritisierte das Fehlen von Zufall und Unberechenbarkeit in den Empfehlungen bei Amazon, Facebook oder iTunes. Gert Loovink protestierte allgemein gegen die "rücksichtslosen Algorithmen" des "Finazismus".

In den letzten Jahren gab es große Fortschritte im "Machine Learning" oder "Deep Learning", und vor allem sind diese Fortschritte im Alltag sichtbar geworden. 2012 verwendeten Sascha Lobo und ich in "Internet – Segen oder Fluch" ein Zitat von Wolfgang Herrndorf: "Den für ihre Technikfeindlichkeit nicht gerade bekannten Japanern ist es bis heute nicht gelungen, einen Go-Rechner zu bauen, der auch nur einen starken Amateur in Schwierigkeiten bringt. Man jubelt stattdessen einem Programm zu, das einen Profi mit neun Steinen Vorgabe versenken konnte. Um in eurem Buch vorzukommen, prophezeie ich hiermit, dass es ein solches Programm auch niemals geben wird, so lange ich lebe. In den folgenden hundert Jahren auch nicht." 2015 schlug die Go-Software AlphaGo den ersten Profi-Spieler, 2017 besiegte sie den Weltranglistenersten.

Es folgt vorsichtshalber eine sehr kurze Erläuterung, worin sich Machine Learning von herkömmlichen Methoden unterscheidet. Wer schon eine Vorstellung davon hat, kann diesen Absatz überspringen. 2008 schrieb ich als Bestandteil eines Twitterbots Code,

---

[1] www.sueddeutsche.de/digital/zur-kritik-an-algorithmen-warum-wurde-mir-ausgerechnet-das-empfohlen-1.1253390, wobei der Titel nicht von mir stammt. Mein Titelvorschlag war "Mama, unter meinem Bett sind Algorithmen".

der dafür sorgen sollte, dass dieser Bot seinen Followern zwar seinerseits zurückfolgen, Spammer aber ignorieren sollte. Er bestand aus mehreren Fragen wie "Enthalten mehr als 17 der letzten 20 Tweets einen Link?", "Sind unter den letzten 20 Tweets keine Antworten, twittert die Person also nur autistisch vor sich hin?", "Enthält die Selbstbeschreibung Wörter wie 'gratis', 'Schnäppchen' oder 'Kredit'?" und dergleichen mehr. Für jedes Ja gab es einen Pluspunkt für Spamverdächtigkeit, für jedes Nein wurde ein Punkt abgezogen. Alle Personen mit 3 oder mehr Pluspunkten wurden als Spammer behandelt. So sieht die traditionelle Methode aus. Beim Machine Learning würde man sich ebenfalls im Vorfeld eine Reihe von Eigenschaften überlegen, die relevant sein könnten, die Gewichtung dieser Eigenschaften aber nicht wie in meinem Code selbst festlegen, sondern durch Ausprobieren ermitteln. Gelingt es mit einer bestimmten Zufallseinstellung der Regler nicht, eine bekannte Spam-Stichprobe von einer bekannten Nicht-Spam-Stichprobe zu unterscheiden, werden die Gewichtungen leicht geändert und dann wieder getestet, ob die Unterscheidung treffsicherer geworden ist. Klappt es mit den bekannten Stichproben, kann man das System auf unbekanntes Material loslassen. (Achtung, grob vereinfachte Darstellung, bitte schreiben Sie auf dieser Basis keine Raketenabwehrsysteme.)

Die Algorithmenkritik hat sich schnell auf dieses Gebiet verlagert. "First Click: Deep learning is creating computer systems we don't fully understand", war im Juli 2016 die Überschrift eines Beitrags im Technologiemagazin "The Verge". Sie suggeriert, dass wir die bisher eingesetzten Computersysteme vollständig durchschauen. In anderen Texten steht noch expliziter, dass diese Durchschaubarkeit erst jetzt nicht mehr gegeben ist, so zum Beispiel im Scientific American, 2016: "... the fact that we no longer fully understand *why* algorithms are deciding things the way they do." Felix Stalder schreibt in seinem ebenfalls 2016 erschienenen Buch "Kultur der Digitalität"[2], diese selbstlernenden Algorithmen seien "in vielen Fällen so komplex, dass sie sich nicht einmal im Nachhinein nachvollziehen lassen. Sie können nur noch experimentell, aber nicht mehr logisch überprüft werden. Solche Algorithmen sind im Grunde Black Boxes, Objekte, die nur über ihr äußeres Verhalten verstanden werden können, deren innere Struktur sich aber der Erkenntnis entzieht."

Das ist alles richtig und auch wieder nicht. Es klingt so, als sei die innere Struktur von Software der Erkenntnis bis gerade eben noch leicht zugänglich gewesen. Die mangelnde Durchschaubarkeit von Softwaresystemen beschäftigt Journalisten und Entwickler aber schon länger.

1987 erschien unter dem Titel "Es ist eine Explosion des Quatsches"[3] im *Spiegel* ein Streitgespräch zwischen den Informatikern Klaus Haefner und Joseph Weizenbaum. Aus diesem Grund lautete der Arbeitstitel meines Beitrags zunächst "30 Jahre Black Box". Haefner war zum Zeitpunkt des Gesprächs seit 18 Jahren im Geschäft, Weizenbaum seit 36. Der moderierende Spiegel-Redakteur Michael Haller sagt darin dasselbe, was Chris Volinsky zwanzig Jahre später über Netflix äußern wird: "Komplizierte Operationen können oft nicht einmal mehr die Programmierer durchschauen, die das fragliche Programm geschrieben haben. Was genau im 'Innenleben' abläuft, bleibt ihnen ein Rätsel. Man hilft sich dann mit dem Denkmodell der "Black box" und fragt nur noch: Welche Resultate kommen aus der Maschine, wenn ich diese Daten eingebe, welche, wenn ich jene eingebe?" Haefner pflichtet ihm bei: "Da sind wir an einem ganz heißen Punkt. Leider, leider werden immer neue Systeme gebaut, die so kompliziert sind, daß

[2] Felix Stalder: "Kultur der Digitalität" (2016)
[3] www.spiegel.de/spiegel/print/d-13520856.html

wir sie nicht mehr durchschauen können. (...) Daß der Mensch unberechenbare Seiten hat, damit konnten wir bislang leben. Ein für uns unberechenbar rechnender Computer, das ist gefährlich und widerspricht der Zielrichtung der Evolution."

Woher nimmt Haller in seinem Kommentar die verwirrten Programmierer und die Black Box? Wahrscheinlich aus Texten von Joseph Weizenbaum, der im gleichen Jahr in der Zeitschrift LOG IN[4] berichtete: "Ich habe mal Karl Deutsch besucht, als ich gerade in Berlin war, und zwar am Steinplatz im Wissenschaftszentrum. Wir haben in seinem Büro ein bißchen geplaudert, und ich habe ihm von der Undurchschaubarkeit großer Systeme erzählt. Ich habe ihm erzählt, daß es überhaupt kein großes Computersystem gibt, das irgend jemand durchschaut, daß es zu spät ist für die Beherrschung solcher Systeme, weil die Systeme eine Konsequenz ihrer Geschichte sind und die Geschichte verlorengeht." Als Beispiel nennt er GLOBUS, ein um diese Zeit in Berlin entwickeltes Simulationsprogramm für die Weltwirtschaft.

Globus bestand aus etwa 10.000 Zeilen Code in Fortran 77. "Das sind riesige Programme", sagt Weizenbaum, "von denen meine ich, daß niemand sie durchschauen kann." Eine durchschnittliche iPhone-App hatte 2013 etwa die fünffache Länge, das Hubble-Weltraumteleskop zwei Millionen Zeilen, der Browser Firefox um die 10 Millionen. Facebook über 60 Millionen, Google insgesamt um die zwei Milliarden.[5] Aber Weizenbaum hat deshalb nicht unrecht: 10.000 Zeilen können undurchschaubar sein. Eine einzige Zeile kann undurchschaubar sein.

Wenig später musste ich meinen Titel in "40 Jahre Black Box" ändern, denn diese Ideen tauchen bereits in Weizenbaums 1976 auf Englisch und 1977 auf Deutsch erschienenem Buch "Computer Power and Human Reason" auf. Es enthält ein ganzes Kapitel über "Unverständliche Programme". "Diese riesigen Computersysteme", heißt es darin, "sind in der Regel von Programmierteams zusammengebastelt worden (man kann es nicht immer 'konstruiert' nennen), deren Arbeit sich oft über einen Zeitraum von vielen Jahren erstreckt. Wenn das System dann endlich gebrauchsfertig ist, haben die meisten der ursprünglichen Programmierer gekündigt oder sich anderen Projekten zugewendet. Gerade dann, wenn solche Systeme schließlich zum Einsatz kommen, ist ihr innerer Ablauf für einen Einzelnen oder ein kleines Team nicht mehr zu verstehen."

Von dort war der Weg zu "50 Jahre Black Box" ein kurzer, denn Weizenbaum zitiert einen 1967 erschienenen Text des Künstliche-Intelligenz-Forscher Marvin Minsky mit dem schönen Titel "Why Programming is a Good Medium for Expressing Poorly Understood and Sloppily Formulated Ideas"[6]. Es geht darin um den "verbreiteten Aberglauben", es sei unmöglich, ein Programm zu schreiben, wenn man nicht extrem präzise formulieren könne, was darin passieren soll und wie. Am Beispiel von Software, die zu diesem Zeitpunkt Menschen im Dame-Spiel besiegen kann, beschreibt Minsky, dass Programmierer, "wenn sie einmal das Anfängerniveau hinter sich gelassen haben, nicht einfach 'Abfolgen von Befehlen' hinschreiben. Stattdessen schreiben sie Einzelaufträge für die Individuen kleiner Gesellschaften, die Prozesse. Egal, wie sehr wir uns bemühen, können wir nur selten alle Details ihrer Interaktionen vorhersehen. Sonst bräuchten wir ja schließlich keine Computer." Die große Illusion, dass das Programmieren eine präzise und starre Ausdrucksform sei, beruhe auf einer Verwechslung von Form und Inhalt. "Es ist vollkommen richtig, dass man sich grammatikalisch sehr präzise ausdrücken muss, wenn das Programm überhaupt laufen

---

[4] www.log-in-verlag.de/PDF-Dateien/Weizenbaum-Interview.pdf
[5] Quelle für alle Zahlen: www.informationisbeautiful.net/visualizations/million-lines-of-code/
[6] Marvin Minsky: "Why Programming is a Good Medium for Expressing Poorly Understood and Sloppily Formulated Ideas" (1967)

soll. Tippfehler oder Zeichensetzungsfehler sind nicht erlaubt! Aber es ist vollkommen falsch, dass man deshalb eine genaue Vorstellung davon hat, was dabei herauskommen wird." Mit dem Wachsen eines Programms verliert der Programmierer die Details aus den Augen und kann nicht mehr vorhersagen, was passieren wird, "er beobachtet das Programm wie ein unberechenbares Lebewesen." Das sei bereits jetzt in manchen großen Programmen so, aber die Ära der Mehrbenutzersysteme beginne gerade erst, und damit werde das Problem akut. Mehrere Programmierer arbeiteten dann gemeinsam an einem Programm, jeder von seinem eigenen Terminal aus, und keiner von ihnen werde es ganz verstehen. "Jetzt erkennen wir das eigentliche Problem mit Aussagen wie 'er macht nur das, was der Programmierer ihm sagt'. Den *einen* Programmierer gibt es gar nicht."

Drei Jahre zuvor war Stanisław Lems "Summa technologiae" erschienen. Ob Weizenbaum oder Minsky das Buch kannten, weiß ich nicht, es wurde erst 1976 ins Deutsche und 2013 ins Englische übersetzt. Es enthält einen Abschnitt "Die Black Box", der mit der Aussage beginnt, kein Einzelner kenne den Aufbau sämtlicher Apparate der modernen Industriegesellschaft. Nur die Gesellschaft als Ganzes besitze diese Kenntnis. Dieser "Prozeß der Entfremdung" werde "vorangetrieben durch die Kybernetik, die ihn auf ein höheres Niveau hebt, denn prinzipiell kann sie auch solche Gebilde hervorbringen, deren Struktur schon keiner mehr kennt. Der kybernetische Apparat wird (die Fachleute benützen gern diesen Ausdruck) zur 'black box'." Die bisher gebauten Black Boxes seien noch so einfach, dass der "kybernetische Ingenieur" noch die Art des Zusammenhangs zwischen den Eingangszuständen und den Ausgangszuständen kennt. "Denkbar ist aber auch eine Situation, in der selbst er den mathematischen Ausdruck dieser Funktion nicht mehr kennt." Denn komplexe Systeme wie "bisher nicht existierende 'sehr große black boxes' … besitzen keine Algorithmen". Lems Begründung dafür: Der Algorithmus muss wiederholbar sein, die Vorhersage künftiger Zustände gestatten, "während ein und dieselbe Gesellschaft, zweimal vor die gleiche Situation gestellt, sich durchaus nicht analog verhalten muß. Und genauso ist es mit allen Systemen von hochgradiger Komplexität."

Am Ende landet man meistens bei Norbert Wiener, dem Begründer der Kybernetik. In Wieners 1960 erschienenem Artikel "Some Moral and Technical Consequences of Automation"[7] dienen wieder Dame-Programme als Beleg: "Das gegenwärtige Niveau dieser lernenden Maschinen sieht so aus, dass sie ein anständiges Amateurschach spielen, aber beim Damespiel nach 10 bis 20 Stunden Arbeit und Indoktrination dem Spieler, der sie programmiert hat, eindeutig überlegen sind." Wer schon einmal gegen sie gespielt habe, müsse zugeben, dass sie originelle Strategien verfolgten. Das widerlege sowohl die verbreitete Annahme, dass Maschinen nichts Originelles hervorbringen könnten, als auch die, dass sie grundsätzlich ihrem Schöpfer unterlegen blieben und seiner vollständigen Kontrolle unterlägen.

Wiener behauptet nicht die Existenz prinzipiell unverständlicher Technik. Er sagt nur, dass das Verstehen zu lange dauert: Bis wir auf die von unseren Sinnesorganen übermittelten Informationen reagieren und auf die Bremse treten können, ist das Auto womöglich schon an eine Wand gefahren. "Es ist gut möglich, dass wir grundsätzlich nicht in der Lage sind, eine Maschine zu bauen, deren Verhaltenskomponenten wir nicht früher oder später verstehen können. Das bedeutet noch lange nicht, dass wir in der Lage sein werden, diese Komponenten in wesentlich kürzerer Zeit zu verstehen, als für den Betrieb der Maschine erforderlich ist, vielleicht lässt sich nicht einmal die Anzahl der dafür nötigen Jahre oder Generationen benennen." Das Begreifen könne der Erledigung

[7] Norbert Wiener: "Some Moral and Technical Consequences of Automation" (1960), www.nyu.edu/projects/nissenbaum/papers/Wiener.pdf

der ursprünglich gestellten Aufgabe weit hinterherhinken. “Das bedeutet auch, dass Maschinen zwar theoretisch der menschlichen Kritik unterliegen, die Kritik aber nicht wirksam wird, weil sie zu spät kommt und nicht mehr relevant ist.” Weizenbaum kommentiert sechzehn Jahre später, das von Wiener hier als Möglichkeit Beschriebene sei inzwischen Realität geworden.

Weizenbaums über vierzig Jahre altes Buchkapitel über unverständliche Software enthält bereits fast alle Formulierungen und Argumente der aktuellen Kritik an Machine-Learning-Software. “Unsere Gesellschaft verlässt sich zunehmend auf Computersysteme, die ursprünglich bei Analysen und Entscheidungen ‘helfen’ sollten, aber längst das Verständnis ihrer Nutzer übersteigen und gleichzeitig unentbehrlich geworden sind. Das ist eine sehr ernste Entwicklung. Sie hat zwei wichtige Konsequenzen. Erstens werden Entscheidungen von Computern beeinflusst und manchmal sogar ausschließlich von Computern getroffen, deren Programme niemand mehr kennt oder versteht. Also kann niemand die Kriterien oder Regeln kennen, auf denen diese Entscheidungen beruhen. Zweitens werden die Regeln und Kriterien, die in diesen Computersystemen verkörpert sind, immun gegen Veränderungen, da ohne detailliertes Verständnis der inneren Abläufe jede nennenswerte Modifikation aller Wahrscheinlichkeit nach das ganze System lahmlegen wird, womöglich irreparabel. Solche Systeme können daher nur weiter wachsen.”

Dass ein Sachverhalt schon länger besteht, ohne dass bisher die Welt untergegangen ist, muss nicht heißen, dass er harmlos ist. Vielleicht geht die Welt ein bisschen später trotzdem unter, oder vielleicht sind wir bereits die gründlich indoktrinierten Produkte dieser Fehlentwicklung, unfähig, das Problem überhaupt noch zu erkennen. Aber jedenfalls greift es zu kurz, die Machine-Learning-Verfahren der letzten Jahre zu beschuldigen und eine Rückkehr zu der einfach und vollständig durchschaubaren Software zu fordern, wie wir sie noch vor fünf Jahren, na gut: vor zehn … oder wenigstens zwanzig … aber doch ganz sicher vor fünfzig Jahren hatten.

Noch einmal kurz zusammengefasst die Gründe für die Unverständlichkeit von Software: Bei Wiener ist es unsere Langsamkeit im Denken. Wir könnten wohl verstehen, was ein Programm tut, es dauert nur unter Umständen so lange, dass die Kritik zu spät kommt. Für Lem verhalten sich Systeme ab einem bestimmten Komplexitätsgrad grundsätzlich unvorhersehbar. Minsky nennt die Interaktionen der einzelnen Prozesse, das Wuchern des Codes und die Auswirkungen der Zusammenarbeit mehrerer Programmierer. Bei Weizenbaum sind es die Größe der Systeme, das Abwandern der ursprünglichen Programmierer und das Vergehen der Zeit, “weil die Systeme eine Konsequenz ihrer Geschichte sind und die Geschichte verlorengeht.”

Ich ergänze zwei letzte Unverständlichkeitsursachen, die in den Zitaten nicht zur Sprache kommen: Die erste sind Bugs, also Fehler in der Soft- und Hardware. Sie waren vor 50 Jahren nicht seltener als heute. Eventuell wissen wir inzwischen ein bisschen genauer, dass sie unvermeidlich sind. Das Problem lässt sich nicht einfach dadurch aus der Welt schaffen, dass die Verantwortlichen besser aufpassen. Die zweite ungenannte Ursache: Code besteht nicht nur aus dem, was sich die einzelne Programmiererin ausdenkt. Auch das einfachste Programm verlässt sich auf eine Menge an Fremdcode, die die Menge des eigenen um Größenordnungen übersteigt: Eingebundene Codebibliotheken und Frameworks, die häufig wiederkehrende Aufgaben lösen, Softwareschichten, die den hingeschriebenen Code in eine Sprache übersetzen, die der Prozessor verarbeiten kann, Software, die das alles anzeigt, speichert oder überträgt. Gelegenheitsprogrammiererinnen wie ich verstehen die obersten ein oder zwei dieser Ebenen. Profis haben etwas mehr Einblick. Eventuell existieren ein paar Menschen, die

die gesamte Software eines Systems bis zur untersten Ebene durchschauen. Aber selbst dann ist immer noch die Hardware für sie opak. Die Black Box ist keine Kiste, die man nur zu öffnen braucht, um den Inhalt unverstellt zu erkennen. Sie enthält weitere Black Boxes. Frühe Programmierer machten sich über die nachrückenden "software engineers" lustig, die die Hardware als gegeben hinnahmen. Die Kritiker waren vermutlich ihrerseits von ihren Vorgängern verspottet worden, weil sie fertige Chips aus dem Versandhandel bezogen, anstatt ihre elektronischen Schaltungen selbst zusammenzulöten. So verlässt sich jeder auf das Funktionieren von Elementen, die für ihn Black Boxes darstellen.

Die meisten dieser Phänomene sind nicht auf Software beschränkt, man kann sie auch an Gebäuden oder technischen Einrichtungen beobachten, die im Laufe der Jahrzehnte umgebaut und an neue Erfordernisse angepasst worden sind. Die Dokumentation ist selten auf demselben Stand oder auch nur am selben Ort wie ihr Gegenstand. Das James Gregory Telescope in St Andrews ist das größte Teleskop Schottlands, ich schreibe Teile dieses Beitrags in seiner Nähe. Die Betriebsfähigkeit des über 50 Jahre alten Geräts hängt im Wesentlichen von einem einzigen pensionierten Informatiker ab, der sich in seiner Freizeit in die verschiedenen historischen Schichten der Teleskoptechnik eingearbeitet hat. Eine Dokumentation der verschiedenen Umbauten existiert – außer im Kopf von Roger Stapleton – nur fragmentarisch und an verstreuten Orten. Das ist keine Ausnahme, sondern der Normalzustand; vom unvollendeten Berliner Flughafen hört man, dass er diesen Zustand der Undokumentiertheit bereits vor seiner Eröffnung erreicht hat.

Ich war mir nicht sicher, ob Chris Volinsky in dem eingangs erwähnten Artikel über Netflix korrekt wiedergegeben worden war. Wir Autoren nehmen es ja manchmal nicht so genau mit dem Zitieren, und Aussagen in indirekter Rede sind doppelt verdächtig. Deshalb fragte ich via Twitter nach und Volinsky antwortete: "It's a fair quote. The final solution was an ensemble of 800+ models. No way to comprehend that fully." Aber was heißt das eigentlich, "comprehend", und "fully"? Was ist mit der immer wieder eingeforderten "Transparenz" oder "Verständlichkeit" von Algorithmen gemeint?

Dafür gibt es unterschiedliche Auslegungsmöglichkeiten. Auf der untersten Ebene geht es darum, ob der Code überhaupt einsehbar ist. Bei Open-Source-Software ist das der Fall, im kommerziellen Bereich meistens eher nicht, und zwar unabhängig von der Frage, ob Machine Learning im Spiel ist. Aber nehmen wir an, der Code sei zugänglich. Ist mit der Transparenzforderung gemeint, dass er für Laien in vertretbarer Zeit durchschaubar sein sollte? Wahrscheinlich nicht, denn das ist bestenfalls bei ganz einfachen Codebeispielen der Fall.

Geht es um die Frage, ob sich das Ergebnis überprüfen lässt? Es gibt Rechenvorgänge, bei deren Ergebnissen das vergleichsweise leicht geht. In einer Einführung in das wissenschaftliche Arbeiten von 1972 heißt es über "programmierbare elektronische Tischrechner" (die nicht nur so hießen, sondern wirklich den halben Schreibtisch belegten): "Wichtig ist aber, daß das Programm auf Fehlerfreiheit kontrolliert wird. Dazu müssen die Ergebnisse, die die Rechenmaschine ausdruckt, mit den Ergebnissen verglichen werden, die man beim Rechnen mit Bleistift und Papier erhalten hat."[8] Aber sobald die Berechnung komplexer wird, kommt man mit Bleistift und Papier nicht mehr weit.

Nehmen wir ein Schachprogramm ohne Einsatz von Machine Learning an, das

[8] H. Seiffert: "Einführung in das wissenschaftliche Arbeiten", 1972

ausschließlich auf einem Entscheidungsbaum mit sämtlichen möglichen Zügen aufbaut. Um der zitierten Empfehlung zu folgen, könnte man diesen Baum mit seinen Verästelungen ausdrucken und dann mit einem Bleistift in der Hand die verschiedenen Zweige durchgehen. Alles, was man braucht, ist sehr viel Druckerpapier und Geduld: Der Baum bildet um die 10^{120} verschiedene Spiele ab. Das bedeutet, dass man zuerst aus jedem der ungefähr 10^{40} Atome im Universum einen erheblichen Stapel Druckerpapier erzeugen müsste und sich dann eine Weile nichts anderes vornehmen dürfte. Eine einfache Machine-Learning-Lösung wäre vergleichsweise leicht zu durchschauen.[9]

In manchen Anwendungsfällen kann man die Korrektheit der Ergebnisse ohne Bleistift überprüfen: Die Wettervorhersage von gestern erweist sich heute als richtig oder falsch. Zu den schwerer überprüfbaren Ergebnissen gehören Empfehlungen von Büchern, Filmen oder Musik. Geschmack ist formbar, und wenn meine Buchhändlerin mir begeistert genug ihre persönlichen Lieblingsromane empfiehlt, werde ich mich vielleicht dafür erwärmen. Wenn Spotify mir ein bestimmtes Musikgenre lange genug vorspielt, überwinde ich womöglich meine Abneigung. Aus einer unpassenden Empfehlung ist eine passende geworden. Die Vorhersage hat die Welt verändert. Und ob die von Schirrmacher bemängelte simulationsbasierte Empfehlung, den Flugverkehr einzustellen, Leben gerettet hat oder überflüssig war, werden wir in Ermangelung eines ansonsten identischen Paralleluniversums nie herausfinden.

Die wichtigste Version der Verständlichkeitsfrage lautet: Lässt sich nachträglich begründen oder sichtbar machen, wie der Weg zum Ergebnis aussah? Die Nachfrage nach solchen Erklärungen ist groß und derzeit wird unter dem Begriff "Explainable Artificial Intelligence" intensiv an ihnen geforscht. Einerseits wäre es eine klare Verbesserung, wenn solche Systeme Auskunft über ihren Lösungsweg erteilen könnten. Andererseits bringt die Erklärung neue Probleme mit sich. Ein simples Beispiel dafür ist der Link "Warum wurde mir das empfohlen?", der seit einigen Jahren die personalisierten Empfehlungen bei Amazon begleitet. Folgt man ihm, dann erhält man zur Antwort, man habe eben vorher Gummistiefel, eine Goethe-Gesamtausgabe und einen Satz Wachsmalkreiden bestellt und deshalb jetzt möglicherweise auch Interesse an diesem Nasenhaarschneider. Das ist nicht immer hilfreich. Und es ist wahrscheinlich eine stark vereinfachte Darstellung dessen, was eigentlich im Inneren der Amazon-Software vorgeht. Das Vorhandensein einer Erklärung beruhigt die Kundschaft, trägt aber wenig zur Aufklärung bei.

Ein 2016 erschienenes Paper, "Generating Visual Explanations"[10], beschreibt Software, die Vogelarten auf Fotos identifiziert und eine textförmige Erklärung dazu liefert: "This is a Kentucky warbler because this is a yellow bird with a black cheek patch and a black crown." Bei genauerem Hinsehen stellt sich allerdings heraus, dass dieser Satz keineswegs den Weg beschreibt, den die Software bei der Identifikation genommen hat. Der Vogel wird durch ein erstes, weiterhin erklärungsloses System bestimmt und der so gefundene Name dann an ein zweites System weitergereicht, das eine für Menschen verständliche Beschreibung aus ihrem Fundus holt. Die Beschreibung erweckt den Eindruck einer Selbstauskunft, ohne wirklich eine zu sein.[11] Vogelbestimmung ist ein harmloses Beispiel. Wenn Unternehmensinteressen oder soziale Erwünschtheit der

---

[9] Eine seriösere Version dieses Arguments findet sich in "The Mythos of Model Interpretability" von Zachary C. Lipton (2017), arxiv.org/pdf/1606.03490.pdf

[10] Hendricks et al., arxiv.org/pdf/1603.08507.pdf

[11] Ich hätte den Taschenspielertrick beim Lesen nicht bemerkt und verdanke den Hinweis diesem Blogbeitrag: blog.foretellix.com/2016/08/31/machine-learning-verification-and-explainable-ai/

Antwort eine Rolle spielen, werden die Betreiber – ob bewusst oder unbewusst – irreführende Erklärungen erzeugen. Das Problem ist danach größer als vorher, weil jetzt auch die Existenz einer Erklärungslücke verschleiert wird.

Weil die Black Box ein altes Problem ist, wird schon länger an Linderungsstrategien gearbeitet. In Flugzeugen kommen seit den 1960er Jahren "Fly-by-Wire"-Systeme zum Einsatz. Die Aktionen der Piloten werden nicht mechanisch oder hydraulisch übertragen, sondern elektronisch übermittelt. Das heißt, dass Software im Einsatz ist, deren Versagen tödliche Folgen haben kann. Deshalb sind sicherheitskritische elektronische Systeme im Flugzeug doppelt, dreifach und manchmal vierfach redundant ausgelegt. Redundanz bedeutet hier nicht, dass ein und dasselbe System mehrmals vorhanden ist, falls eins davon kaputtgeht. Es ist unterschiedliche Software, die zum Teil auf unterschiedlicher Hardware läuft und von separaten Teams entwickelt wird. In Tests müssen die Systeme bei identischen Eingaben zu denselben Ergebnissen gelangen. Dass ein bestimmtes Problem – ein Fehler in der Hardware, ein Fehler in der Software, eine Fehler durch ungünstige Umweltbedingungen – alle diese Systeme auf die gleiche Weise in die Irre führt, ist nicht vollständig unmöglich, aber unwahrscheinlich. In einem dreifach redundanten System können zwei Elemente, die zum gleichen Ergebnis kommen, ein drittes abweichendes Element überstimmen. (Aus dem gleichen Grund wurde in der Seefahrt im 19. Jahrhundert die Mitnahme von mindestens drei Chronometern empfohlen[12]; die HMS Beagle hatte auf Darwins Reise 22 Stück an Bord.)

Software läuft auf Hardware, und sowohl Mikroprozessoren als auch Platinen werden schon seit den 1970er Jahren nicht mehr in Handarbeit, sondern wiederum von Software designt ("Electronic Design Automation"). Nur dadurch ist es möglich, auf einem aktuellen Mikroprozessor mehrere Milliarden Komponenten unterzubringen. Es gibt keinen Menschen, der nachzuvollziehen oder mit dem Bleistift zu überprüfen versucht, ob diese Komponenten auch korrekt verbaut sind. Dafür sind weitere Softwaresysteme zuständig.

In der Flugzeugelektronik wie im Hardwaredesign sind die Systeme komplex und nicht in vertretbarer Zeit vollständig durchschaubar. Die Strategien zum Umgang mit dem Problem bestehen nicht im Ruf nach "transparenten" Systemen oder der Forderung, auf ihren Einsatz zu verzichten. Sie umfassen die Entwicklung neuer Hilfswerkzeuge zur Visualisierung des Geschehens, automatisierte Testverfahren und den Abgleich mehrerer Ergebnisse aus unterschiedlich arbeitenden Systemen.

Es gibt eine ganze Reihe von Software-Testmethoden, die keinen Einblick ins Innenleben des Systems erfordern und unter dem Namen "Black-box testing" zusammengefasst werden. *Black-box testing* ist es, wenn man vor dem Kauf eines Gebrauchtwagens nur einen flüchtigen Blick unter die Motorhaube wirft und statt der Untersuchung sämtlicher Bauteile lieber auf einer Probefahrt überprüft, ob das Auto das kann, was Autos können sollten: Fahren, beschleunigen, bremsen, schalten, keine seltsamen Geräusche machen. Allerdings unternimmt man in der Informatik nicht nur eine Probefahrt, sondern sehr viele, und testet dabei auch seltene und unvorhergesehene Fälle: Was passiert, wenn

---

[12] Wikipedia: "The Admiralty started a general issue of chronometers to H. M. Ships from 1825, but between about 1800 and 1840 the availability of chronometers could not keep up with the demand. The Admiralty therefore only issued one chronometer to each ship unless the Captain personally owned one. In those cases the Admiralty would issue a second machine to make the total up to three, reasoning that a ship with two was no better off than with one since in the event of a discrepancy it was not possible to identify the faulty instrument."

das Auto eine senkrechte Wand hochzufahren versucht?

Black-box testing ist nicht nur eine Technik für Situationen, in denen man tatsächlich keinen Einblick in den Code hat. Das Verfahren wird auch bei Software eingesetzt, die offen daliegt, weil es Vorteile bietet. Durch Black-box testing lassen sich pro Zeiteinheit insbesondere bei großen, komplexen, unscharfen Aufgabenstellungen mehr Fehler aufdecken als durch den Versuch der logischen Überprüfung, wie sie Felix Stalder im eingangs zitierten Text vermisst.[13] Der Robotikforscher Rodney Brooks nennt die mathematische Beweisführung in Bezug auf Künstliche Intelligenz grundsätzlich eine vergebliche Hoffnung: "Es gelingt uns nicht, mit großen Teams in mehrjähriger Arbeit zu beweisen, dass ein 1000 Zeilen langes Programm gegen Hackerangriffe abgesichert ist, also werden wir im Zusammenhang mit großen AI-Systemen ganz sicher nicht viel beweisen können."[14]

Bei den Anwendungen von Machine Learning, die derzeit besonders kritisch diskutiert werden, kommen redundante Systeme oder andere Linderungsstrategien noch kaum zum Einsatz. Das liegt – abgesehen davon, dass das ganze Forschungsgebiet recht neu ist – an zwei unvorteilhaften Eigenheiten: Erstens betreffen die Folgen in vielen Bereichen nur Einzelpersonen, die nicht auf den ersten Blick erkennbar gemeinsame Opfer eines versagenden Systems sind wie beim Absturz eines Flugzeugs. Zweitens gibt es häufig keine Konkurrenz. Wenn meine Bank sich idiotisch verhält und als Begründung vorbringt "Die Software gibt es so vor, daran können wir auch nichts ändern", wechsle ich die Bank. Wenn die Flugzeuge der Fluggesellschaft A ständig vom Himmel fallen, werden Passagiere lieber mit Fluggesellschaft B fliegen. Aber diese Möglichkeit existiert nicht überall. Bürger können nicht so leicht den Staat wechseln, Häftlinge sich nicht aussuchen, welche Software ihre Rückfallwahrscheinlichkeit beurteilen soll, und auch die Schufa gibt es nur ein einziges Mal. Die Vereinzelung der Fälle und der Mangel an Konkurrenz führen dazu, dass es für die Hersteller weniger Anreize gibt, auf Maßnahmen zur Fehlerreduzierung zu setzen.

Die Alternative zu den kritisierten Systemen besteht meistens darin, dass ein Mensch darüber entscheidet, ob jemand vorzeitig aus der Haft entlassen werden, einen Kredit, eine medizinische Diagnose oder einen Versicherungsvertrag bekommen soll. (Solche Entscheidungen werden schon seit Jahrzehnten unter Zuhilfenahme von Software getroffen, die weder für ihre Anwender noch die von der Entscheidung Betroffenen sonderlich transparent ist. Aber dazu sage ich jetzt nichts, sonst werden wir hier nie fertig.) In einem 2016 erschienenen Paper über "The ethics of algorithms"[15] heißt es: "Die algorithmische Verarbeitung steht im Gegensatz zur traditionellen Entscheidungsfindung, wo menschliche Entscheider auf Befragen ihre Gründe benennen können, beschränkt nur durch die Bereitschaft und Fähigkeit der Entscheider, eine Erklärung abzugeben, und die Fähigkeit der Fragesteller, sie zu verstehen." Diese Aussage hat ein großes Problem. Es liegt in dem Wort *Fähigkeit*. Wir lassen uns in unseren Entscheidungen von einer Vielzahl von Faktoren beeinflussen, die uns nicht vollständig bewusst sind, das ist ein alter und gründlich erforschter Hut. Wenn eine Buchhändlerin eine Empfehlung erteilt, ist der Weg zu dieser Empfehlung unklarer als bei jedem Empfehlungsalgorithmus. Abgesehen von Sonderfällen wie "Davon haben wir versehentlich eine Palette zu viel bestellt, das empfehle ich so lange allen Kunden, bis es weg ist" wird die Buchhändlerin die Gründe für ihre Empfehlung nicht benennen

[13] Mehr dazu: blog.foretellix.com/2015/07/14/fv-has-much-better-pr-than-cdv/
[14] rodneybrooks.com/the-seven-deadly-sins-of-predicting-the-future-of-ai/
[15] Mittelstadt et al.: "The ethics of algorithms: Mapping the debate" (2016)

können. Sie wird zwar eine Erklärung vorbringen, aber der Zusammenhang zwischen dieser Auskunft und den tatsächlichen Hintergründen der Empfehlung ist unklar. Das menschliche Gehirn ist eine Black Box, nicht nur von außen betrachtet, sondern auch für seine Besitzerin.

Darauf weist auch Stanisław Lem in seinem "Summa Technologiae"-Kapitel hin. Dass es grundsätzlich möglich sei, Black-Box-Systeme zu konstruieren, ohne sie zu planen oder zu durchschauen, wüssten wir, weil wir selbst Black Boxes seien. "Demnach ist also jeder Mensch ein hervorragendes Beispiel eines Apparats, dessen wir uns bedienen können, ohne seinen Algorithmus zu kennen. Einer der 'uns am nächsten stehenden Apparate' im ganzen Kosmos ist unser eigenes Gehirn, denn wir haben es ja im Kopf. Trotzdem wissen wir bis zum heutigen Tage nicht genau, wie dieses Gehirn funktioniert. Seine Mechanismen introspektiv zu erforschen, ist, wie die Geschichte der Psychologie zeigt, in höchstem Maße trügerisch und führt zu den allerverkehrtesten nur denkbaren Hypothesen." Dass die Natur solche Black Boxes hervorgebracht hat, zeige aber, dass ihre Konstruktion möglich sei und ihre Undurchschaubarkeit dem Funktionieren nicht grundsätzlich im Weg stehe.

Erklärungslücken halten uns nicht davon ab, unser Gehirn zu benutzen. Sie halten uns auch nicht davon ab, Vollnarkosen einzusetzen, obwohl bis heute unbekannt ist, warum und wie sie wirken. Die Vorteile der Narkose sind so groß, dass wir bereit sind, über den Nachteil ihrer Rätselhaftigkeit hinwegzusehen. Ich möchte die Undurchschaubarkeit von Gehirn, Narkose und Software nicht verteidigen, sondern nur dafür plädieren, an Software keine strengeren Maßstäbe anzulegen als an die anderen beiden Black Boxes. Dass die Verwendung bereits läuft, obwohl eine vollständige Erklärung noch aussteht, ist nicht ungewöhnlich. Zitronen als Mittel gegen Skorbut waren jahrhundertelang im Einsatz, bevor geklärt wurde, warum und wie sie wirken.

Jessa Jones, Betreiberin einer Reparaturwerkstatt für die Mikroelektronik in Apple-Geräten, erläutert in einem Video[16] ihre Arbeitsweise: "Von Seiten des Herstellers gibt es keinen Support für Reparaturen. Wir müssen alles selber rausfinden. Ich komme ursprünglich aus der Molekulargenetik, und da geht es genau darum, wie man winzige Dinge versteht, die man nicht sehen kann und für die es keine Gebrauchsanweisung gibt." Reverse Engineering und Biologie sind zwei Beispiele für Arbeitsfelder, in denen Forschende regelmäßig Einblick in komplexe, intransparente und nichtlineare Systeme gewinnen. Insbesondere in der Biologie kommt man mit Forderungen nach transparenteren, besser dokumentierten, leichter durchschaubaren Lebewesen nicht weit. Auch transparente Tiere erleichtern die Arbeit nur unwesentlich. Dass Erkenntnisgewinn trotzdem möglich ist, verdanken wir einer Sammlung von Strategien und Methoden. Sie heißt Wissenschaft und hat sich bisher eigentlich ganz gut bewährt.

[16] YouTube: "The Master Microfixer Teaching the World to Fix iPhones"