Algorithmen erklärt Euch!

Methodische Überlegungen zum nutzerzentrierten Kuratieren KI-basierter Entscheidungssysteme am Beispiel von Routenplanern

Annelie Pentenrieder

Der Alltag ist durchdrungen von digitalen, algorithmen-basierten Empfehlungssystemen, die als Entscheidungsfilter für verschiedene Angelegenheiten dienen. Algorithmen geben Kaufempfehlungen, sie sortieren Informationen in Suchmaschinen und News-Feeds oder wählen eine passende Route im Straßenverkehr aus. Damit sind Algorithmen zu grundlegenden Alltagshilfen geworden, doch ihre berechneten Handlungsvorschläge zu hinterfragen, fällt schwer.[1] Je nachdem, für welche Zwecke eine Empfehlung mittels Algorithmen eingeholt wird, gibt es immer wieder Situationen, in denen es sinnvoll sein kann, die »Grammatik« (Seaver 2019) zu kennen, nach denen algorithmische Empfehlungen zustande kommen. Ein solcher Einblick in die Entwicklung von Entscheidungsalgorithmen ist für Nutzer*innen derzeit jedoch nicht möglich. Obwohl Algorithmen das Handeln in digitalisierten Alltags- und Arbeitswelten heute in vielfältiger Weise beeinflussen, ist die Art und Weise dieser Beeinflussung undurchdringlich und nicht aushandelbar. Mit Algorithmen, wie sie aktuell konstruiert werden, bleibt unklar, wer aus welchen Gründen den ein oder anderen Handlungsvorschlag erhält. Algorithmische Entscheidungssysteme verteilen Wissen neu zwischen denen, die sie nutzen, und denen, die sie programmieren und verkaufen. Diese Umverteilungen gilt es öffentlich zu verhandeln und auszuhandeln. Für eine demokratische Technikgestaltung müssen Algorithmen erklärbar werden, das bedeutet, dass unterschiedlichen Gruppen, wie etwa den Nutzer*innen selbst aber auch Vertreter*innen des Verbraucherschutzes, sozialwissenschaftlichen Forschungsansätzen oder juristischem Personal eine

1 Algorithmen verstehe ich mit Bezug auf Angèle Christin (2020) und Solon Barocas et al. (2014) als Softwareprogramme, die computergestützte Aufgaben auf Basis digitaler Daten ausführen. Algorithmen sind Sequenzen von logischen Operationen, die für den Rechner Schritt-für-Schritt-Instruktionen bereitstellen, um die Daten zu bearbeiten.

Interpretation von Algorithmen ermöglicht werden muss. Dieses Interpretationsdesiderat von Algorithmen beschäftigt zahlreiche Forschungen in den Computer- und Sozialwissenschaften (Burrell 2016; Seyfert 2017; Spielkamp 2019; Zweig 2019; Christin 2020; Pentenrieder 2020). Gleichzeitig entstehen erste Ansätze zur Erklärbarkeit von Algorithmen: Sandra Wachter schlägt am Beispiel von Bankdarlehen das Konzept der »kontrafaktischen Erklärbarkeit« vor: Wenn eine Kundin auf Basis einer algorithmischen Berechnung keinen Kredit von ihrer Bank erhält, interessiert sie sich nicht dafür, wie der Algorithmus technisch funktioniert, sondern sie will wissen, an welchen Eigenschaften ihres Kundenprofils es gelegen hat, dass sie den Kredit nicht erhalten hat (Wachter 2019). Eine andere Art der Erklärung algorithmischer Entscheidungsprozesse zeigt Nick Seaver für die Nutzer*innen des Musikstreaming-Dienst Pandora. Basierend auf mehreren Hundert Klangattributen, die Musikexpert*innen zuvor zusammengestellt haben, begründet der Musikstreaming-Dienst seinen Kund*innen die Auswahl der nächsten Musikstücke: »Based on what you've told us so far, we're playing this track because it features latin influences, danceable beats, a rhythmic intro, the use of chordal patterning and use of tonal harmonies« (Seaver 2019: 2).

Diese beiden Beispiele aus dem Bankenwesen und aus dem Musik-Streaming zeigen, dass jede Situation und jedes Anliegen einer anderen Erklärung bedürfen. Wem die Erklärungen zu Algorithmen letztendlich nutzen, hängt auch davon ab, wer an der Interpretation von Algorithmen beteiligt wird und welche Form der Interpretation gewählt wird. Für die Erklärungen von Algorithmen gilt damit dasselbe wie für die Programmierung der Algorithmen selbst: Es bedarf eines anerkannten und ausführlichen Gestaltungs- und Kuratierungsprozesses, um situativ passende Konzepte für algorithmische Erklärbarkeiten zu erstellen.

Am Beispiel von Routenplanern für den Straßenverkehr skizziere ich im Folgenden einen methodischen Vorschlag zur Erforschung von Algorithmen und ihren Erklärbarkeiten. Gerade weil Routenplaner bereits seit zwei Jahrzehnten selbstverständlich gewordene Entscheidungshilfen sind, wird hier der algorithmische Einfluss auf alltägliche Entscheidungssituationen besonders deutlich: Aus vielen ähnlich guten Verbindungen wählen Routenplaner einzelne Routenoptionen aus und regeln Verkehrsflüsse im Straßenverkehr. Mit ihren berechneten Verbindungslinien definieren sie Zugänge und schaffen Barrieren zwischen Orten und Räumen. Zunehmend personalisiert und aus den Nutzerdaten »lernend« empfehlen sie den Fahrer*innen auf Basis bestimmter Profile unterschiedliche Wege zu gleichen Zie-

len.[2] Auch im Straßenverkehr sind somit räumliche »Filterblasen« (Pariser 2012) in Zukunft vorstellbar und Routenvorschläge somit diskussionswürdig.

Das erste Kapitel des Beitrags widmet sich den Algorithmen in Routenplanern und klärt die Rolle der Nutzer*innen in der Interaktion mit KI-Systemen. Das zweite Kapitel skizziert eine methodische Vorgehensweise, Algorithmen in Bezug auf ihre Erklärbarkeiten zu erforschen: Erstens werden Erklärungen schrittweise aus der Perspektive von Nutzer*innen entwickelt. Zweitens dienen architektonische Metaphern dazu, nicht nur die technischen Logiken, sondern auch die Opazität von Algorithmen als relevanten Aspekt für eine Erklärbarkeit zu analysieren. Das dritte Kapitel nennt drei unterschiedliche Einsatzgebiete, in denen algorithmische Erklärbarkeiten eingesetzt werden könnten.

Wegfindungsalgorithmen und die Verheißung von KI

Wie viele andere Alltagsalgorithmen ist der Routenplaner ein Ergebnis der Forschung zu Künstlicher Intelligenz, die aktuell durch die neuen technischen Möglichkeiten des maschinellen Lernens breite öffentliche Aufmerksamkeit erhält. Künstliche Intelligenz (KI) verstehe ich als einen umkämpften, weil verheißungsvollen Überbegriff für eine Vielzahl an technischen automatisierten Systemen, die mit komplexen Algorithmen, vielfältigen Parametern und großen Datenmengen menschliche Entscheidungen lenken, prognostisch unterstützen oder Entscheidungen direkt automatisiert treffen. Bereits Joseph Weizenbaum, einer der KI-Gründungsväter, wies auf die Ambivalenz dieses Begriffs hin:

> »Ich bin der Ansicht, daß ein in jeder Beziehung vereinfachter Begriff von Intelligenz sowohl das wissenschaftliche wie das außerwissenschaftliche Denken beherrscht hat, und daß dieser Begriff zum Teil dafür verantwortlich ist, daß es der perversen, grandiosen Phantasie der künstlichen Intelligenz ermöglicht wurde, sich derart zu entfalten. Ich behaupte dagegen, daß ein Organismus weitgehend durch die Probleme definiert wird, denen er sich gegenübersieht. Der Mensch muß Probleme bewältigen, mit denen sich keine Maschine je auseinandersetzen

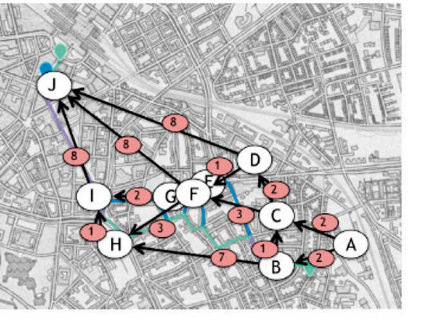
2 Routenalgorithmen gestalten das aktuelle Verkehrsgeschehen täglich mit. Das macht der Künstler Simon Weckert in seinem Berliner Google Maps Hack sichtbar. Mittels künstlich erzeugter Staudaten beeinflusst er die Routenberechnungen auf Google Maps. Mit einem Bollerwagen, der 90 Smartphones geladen hatte, ging er dazu auf einer Berliner Brücke auf und ab. Die öffentliche Resonanz, die dieses Projekt erfuhr, zeigt wie das Interesse an der Machart von Alltagsalgorithmen über öffentlichkeitswirksame Maßnahmen geweckt werden kann. Siehe https://www.sueddeutsche.de/digital/google-maps-hacks-stauanzeige-1.4784081. Zugegriffen: 10. Februar 2020.

> muß, die von Menschenhand gebaut wurde. […] Computer und Menschen sind nicht verschiedene Arten derselben Gattung.« (Weizenbaum 2008: 269)

Folgt man Weizenbaums Verständnis, Technologien von den Problemen her zu definieren, die sie lösen, bleibt der Routenplaner nicht länger ein kompaktes KI-System sondern zerfällt in eine Vielzahl unterschiedlicher und zum Teil klar definierbarer Komponenten und Mechanismen, die alle in unterschiedlicher Weise dazu beitragen, dass eine Route im täglichen Verkehrsgeschehen algorithmisch bestimmbar wird. In Routenplanern überlagern sich algorithmische Mechanismen aus verschiedenen Forschungsepochen der KI-Forschung – von Expertensystemen bis hin zum maschinellen Lernen. Jeder Mechanismus hat dabei für das Routenergebnis seine ganz eigenen Effekte. Den meisten Routenplanern liegt für die grundsätzliche Routenberechnung der »A*-Algorithmus« (siehe Abbildung 1) zugrunde, der die Berechnung des optimalen Weges zwischen zwei Orten ausführt. In den 1960er Jahren als regelbasiertes Expertensystem für die Bewegungen von *Shakey the Robot* am Standford Research Lab entwickelt, prüft dieser Algorithmus in einer nachvollziehbaren, weil stets unveränderten Iteration, über welche Straßen die kostengünstigste Route berechnet werden kann. Für die zu prüfenden Straßen liegen dem Algorithmus Kostenwerte in einem Graphen (siehe Abbildung 2) vor. Der Graph ist das mathematische Modell, in dem das Straßennetz repräsentiert wird. Im Algorithmus werden die Kostenwerte der einzelnen Straßen addiert und die Route ausgewählt, die gemäß des geltenden Kostenprofils die geringsten Gesamtkosten aufweist.

*Abbildung 1: A*Algorithmus; Abbildung 2: Straßennetz im Graph modelliert (orientiert an Velden 2014)*

```
1  begin
       // INITIALISIERUNG
2      d(v[A]) ← 0; parent(v[A]) ← v[A]
3      foreach i ∈ {B, C, D, E, F, G, H, I, J} do
4          d(v[i]) ← ∞; parent(v[i]) ← NULL
5          h(v[i]) ← √(Σ(v[i].coord − ziel.coord)²)
       // ITERATION
6      while queue ≠ ∅ do
7          u ← queue.extractMin()
8          if u == ziel then
9              return "Weg gefunden"
10         foreach (u, w) ∈ E do
11             dist ← d(u) + l(u, w)
12             if w ∈ queue and d(w) > dist then
13                 d(w) ← dist; f ← dist + h(u); parent(w) ← u
14             else if parent(w) == NULL then
15                 d(w) ← dist; f = dist + h(u); parent(w) ← u
16                 queue.insert(w, f)
17     return "Ziel nicht erreichbar"
```



Der A*-Algorithmus wird in heutigen KI-Forschungsabteilungen um adaptive Verfahren wie das maschinelle Lernen ergänzt. Die »Kostenwerte« von Straßen werden dadurch nicht mehr nur darüber definiert, wie schnell auf einem bestimm-

ten Straßentyp (z.B. Landstraße versus Autobahn) in Bezug auf die dort geltenden Geschwindigkeitsbegrenzungen gefahren werden kann. Stattdessen oder ergänzend dazu werden tatsächlich gefahrene Geschwindigkeitswerte auf einer Straße kontinuierlich über die Fahrerdaten der Nutzer*innen von Routenplanern erhoben und aktualisiert, um auch Stau-Wahrscheinlichkeiten zu prognostizieren (Sonnenburg 2019).

Da sich in Routenplanern unterschiedlich komplexe Algorithmen neuerer und älterer KI-Forschungsgenerationen verweben, können an diesem bekannten Alltagsalgorithmus technische Sachverhalte von KI-Systemen modellhaft sichtbar gemacht werden. Die Unterscheidung dieser ineinander verwobenen Softwarekomponenten ermöglicht es wiederum, algorithmische Berechnungsergebnisse auf einzelne technische Gegebenheiten konkret zurückzuführen. Eine solche Analyse wirkt Mythen entgegen, dass KI-Systeme nicht zu öffnende Blackboxes seien, weil sie grundsätzlich zu komplex sind, um von Nutzer*innen verstanden zu werden. Eine technisch-begründete Unterscheidung zwischen verschiedenen Arten von Algorithmen zeigt, so auch Jenna Burrell, dass nicht alle heute genutzten Alltagsalgorithmen zwangsläufig selbstlernend im Sinne von dynamischen, adaptiven oder statistischen Verfahren sein müssen (Burell 2016: 3). Insbesondere solche Algorithmen, die bestimmte Politiken in den Alltag einschreiben, beruhen auch heute noch zum Großteil auf klar definierbaren Rechenschritten und können selbst mit einfachen Mitteln verstanden werden, so Burrell (2016: 3). Die Unterscheidung von Algorithmen mit unterschiedlichen Entstehungsgeschichten und technischen Hintergründen in ein und derselben Technologie, ermöglicht einen »vergleichenden Ansatz« (Christin 2020: 11/12) wie er in der ethnografischen Forschung der STS eine lange Tradition hat (Knorr Cetina 1999; Latour 2010). Der Vergleich unterschiedlicher Algorithmen ermöglicht es, unterschiedliche Programmierkulturen herauszuarbeiten und Spezifika zu konkretisieren, die etwa regelbasierte Systeme von maschinellen Lernverfahren unterscheiden. Auch die Opazität dieser verschiedenen Algorithmen ist unterschiedlich beschaffen und unterschiedlich bedeutend für ihre Nutzer*innen, sodass die Gestaltung von Erklärbarkeiten ebenfalls einer je eigenen Herangehensweise bedarf.

Mit der Möglichkeit zur Nachvollziehbarkeit von KI-Systemen rückt die Frage in den Fokus, welche Aspekte von Software überhaupt offengelegt werden müssen, damit algorithmische Empfehlungen informierter genutzt werden können. Um dies am Beispiel von Routenplanern zu erforschen, habe ich informierte und interessierte Nutzer*innen in die Suche nach relevanten technischen Logiken einbezogen und dazu Lucy Suchmans über 30 Jahre alte Studie zu Mensch-Maschine-Interaktionen für heutige Alltagsalgorithmen reinterpretiert (Suchman 2007). Suchman entwickelte in den 1980er Jahren am Forschungslabor Xerox PARC im Silicon Valley anhand damaliger KI-Systeme einen ethnografischen Ansatz zur Erforschung von Nutzerpraktiken. Sie betont ebenso wie Weizenbaum, dass

Mensch und Maschine jeweils unterschiedlicher Analysekategorien bedürfen und macht einen methodischen Vorschlag, der diese Differenz auch bei der Nutzung von KI-Systemen beibehält:

> »I would propose that the price of recognizing the agency of artifacts need not be the denial of our own. Now that the agencies of things are well established, might we not bring the human out from behind the curtain, so to speak, without disenchantment? This requires, among other things, that we acknowledge the curtain's role. Agencies – and associated accountabilies – reside neither in us nor in our artifacts but in our intra-actions. The question […] is how to configure assemblages in such a way that we can intra-act responsibly and generatively with and through them.« (Suchman 2007: 285)

Um in Suchmans Sinne auch die Kompetenzen zu berücksichtigen, die Nutzer*innen in algorithmisch gestaltete Entscheidungssituationen einbringen, ist entscheidend, wie Nutzerpraktiken in der Interaktion mit KI-Systemen repräsentiert werden. Praxistheoretische Vorgehen bieten darum große Potenziale, wenn es um die Frage nach dem Zuschnitt und der Auswahl von nutzerzentrierten Erklärungen zu Algorithmen geht. Ein solches Vorgehen lege ich im nächsten Kapitel dar.

Softwarekomponenten über Nutzerpraktiken erschließen

Ethnografisches Vorgehen zur Erforschung von Wegfindungsalgorithmen

Im Folgenden schlage ich ein Vorgehen in drei Schritten vor, um Algorithmen auf mögliche Erklärbarkeiten hin zu beforschen. Es basiert auf meinen eigenen Erfahrungen in der ethnografischen Erforschung von Routenplanern entlang der Grounded Theory und ist zunächst »durch ein stetiges Hin und Her zwischen konkreten Forschungsgegenständen, analytischen Kategorien und der eigenen, verkörperten Erfahrung« gekennzeichnet (Star 2017: 13; Strübing 2002). Die nachträgliche ›Verstrukturierung‹ in drei systematisch abgegrenzte Analyseschritte folgt dem Aufruf ethnografischer Forscher*innen wie Angèle Christin (2020) und Nick Seaver (2017), Strategien und Taktiken zur Erforschung von Algorithmen gemeinsam zu sammeln und zu strukturieren. Erstens lassen sich dadurch die Methodiken ethnografischer Algorithmenforschung weiter schärfen und zweitens legt eine solche Sammlung das Potenzial ethnografischer Forschung auch außerhalb der Sozialwissenschaften dar:

> »A central value of contemporary ethnographic research is to try to make explicit as much of the research process as possible for the ethnographic community as a whole. Thus, it is important to document and reflect on the choices, values, and

shortcuts that shape one's relationship to the field. […] Overall, the goal is to come up with a more structured and deliberate methodological toolkit in order to approach these complex objects.« (Christin 2020: 17)

Interessierte Nutzerpraktiken als Ausgangspunkt

Zum Auffinden und Auswählen von Softwarekomponenten des Routenplaners, die einer Erklärung für die Nutzer*innen bedürfen, habe ich zunächst die komplexen Bemühungen von Nutzer*innen untersucht, mit teils unbewussten Praktiken, die ich »Plausibilisierungsstrategien« (Pentenrieder 2020) nenne, operative Logiken ihrer Routenplaner ausfindig machen. Dazu befragte ich Taxi-, Kurier- und Fernfahrer*innen als prototypische, informierte Nutzer*innen. Mittels langjähriger Berufserfahrung und Ortskenntnis prüfen sie algorithmische Empfehlungen im Straßenverkehr auf ihre Plausibilität. Die Untersuchung dieser Berufsgruppe hat den methodischen Vorteil, dass die Fahrer*innen durch ihren Arbeitskontext Routenberechnungen mit eigenem Wissen und Reverse-Engineering-Methoden in besonderer Weise hinterfragen, nicht nur um eine gute Entscheidung zu treffen, sondern auch um ihre Routenentscheidungen gegenüber Vorgesetzten, Kolleg*innen oder Kund*innen begründen zu können.[3] Gerade ein solches Erfahrungswissen kann zentrale Denkanstöße für eine demokratischere Softwaregestaltung liefern.

Die Fragen interessierter Nutzer*innen von Alltagssoftware an den Anfang der Softwareanalyse zu stellen, gewährleistet, dass die Nutzer*innen die Interpretationen der Software liefern, die im nächsten Schritt die Erklärungen erzeugen. Diese Setzung ist zentral, da der Prozess des Interpretierens flexibel ist, so Nick Seaver in Bezug auf Clifford Geertz (Seaver 2019: 7). Es spielt darum eine Rolle, wer die Aufgabe der Interpretation von Software zuerst übernimmt – die Nutzer*innen oder die Entwickler*innen. Je nachdem, wer beim Interpretationsprozess von Algorithmen federführend ist, entstehen jeweils andere Erklärungen.

Um algorithmische Empfehlungen informiert zu nutzen, vollziehen manche Nutzer*innen die programmierten Strukturen nach und überlegen, was sich die Entwickler*innen bei der Programmierung gedacht haben könnten. Solche Überlegungen rücken bestimmte technische Komponenten in den Fokus. Zwei kurze Beispiele sollen dies verdeutlichen: Wenn eine Fahrerin an einer Straßenkreuzung der vorgeschlagenen Richtung ihres Routenplaners nicht folgt, weil sie vermutet, der Routenplaner könne eine neue Straße aufgrund des veralteten Kartenmaterials noch nicht als bessere Verbindung kennen, dann spiegeln sich in ihren Plausibilisierungsstrategien zentrale organisatorische und technische Hintergründe der

3 Ähnliches konnte auch Alex Rosenblat (2018) in ihrer Studie zu Uber-Fahrer*innen beobachten.

Softwareprogrammierung wider. Die Konstruktion des Kartenmaterials und Fragen zu Update-Verläufen haben demnach für ihre Nutzungssituation Relevanz und rücken für eine genauere technische Analyse in den Fokus. Auf eine andere Softwaredynamik macht folgendes Beispiel aufmerksam: Ein Essenskurier, der eine Auftrags-App der Gig-Economy verwendet, fährt bewusst langsam, weil er vermutet, dass seine langsamen Geschwindigkeiten zu kürzeren und damit zu lukrativeren Folgeaufträgen führen könnten (Töpfer 2016).[4] Seine Vermutungen machen die adaptive Datenerhebung und das Tracking von Fahrerdaten als Interaktionsfläche sichtbar. Beide Mensch-Maschine-Interaktionen machen auf technische Schnittstellen aufmerksam, die nicht von den Entwickler*innen, sondern von den Nutzer*innen als Schnittstellen definiert werden. Der Fokus dieser informierten Nutzer*innen richtet sich nicht auf offiziell designte Interaktionsflächen wie etwa Bedienoberflächen oder Nutzereinstellungen, die die Nutzer*innen direkt über Touchscreens oder vorgegebene Tasten erreichen. Die hier angesprochenen Interaktionsflächen liegen stattdessen entlang von technischen Logiken und Dynamiken einzelner Softwarekomponenten und -mechanismen im Hintergrund des Displays. Die Vermutungen der Nutzer*innen geben Anhaltspunkte, welche Aspekte der Software im nächsten Schritt offengelegt werden müssten. In diesem praxistheoretischen Ansatz wird für die Beschreibung von Software ein analytischer Umweg über aktiv interagierende Nutzer*innen genommen. Algorithmen oder Software werden ausschließlich in Bezug zur Nutzererfahrung untersucht. Die Erklärungshoheit bleibt damit zunächst bei den fragenden Nutzer*innen und beginnt nicht bei den Entwickler*innen, die das System konzipieren.

Vignetten als Verbindung von Wissensbeständen

Aus den gesammelten Feldnotizen und Interviewtranskripten der Nutzerstudien erstellte ich im zweiten Schritt »Vignetten«.[5] Dabei handelt es sich um kurze Beschreibungen zu konkreten Interaktionen wie die oben genannten, um den Einfluss von Software anhand konkreter Situationen zu verschriftlichen. Im Zentrum stehen alltägliche Interaktionen, Reverse-Engineering-Praktiken aber auch Fälle, in denen es zu Irritationen zwischen Software und Nutzer*innen kam. Vignetten unterstützen erstens die Auswahl von Softwarekomponenten, die für das algorithmisch berechnete Ergebnis einen Unterschied machen. Zweitens dient dieser Schreibprozess zum Finden von Begrifflichkeiten, die die Softwarekomponenten aus der Logik und Perspektive der Nutzung heraus beschreiben, zunächst noch,

4 Für eine genauere Beschreibung und Analyse der Reverse-Engineering-Verfahren des Fahrradkuriers siehe Pentenrieder 2020.

5 Diese Methodik wird häufig in den Software Studies verwendet. Siehe etwa Kitchin 2007 oder Seaver 2019.

ohne auf das Vokabular der Entwickler*innen zurückzugreifen. Vignetten verbinden die getrennten Wissensbestände von Nutzer*innen und Entwickler*innen und dienen der Übersetzung zwischen dem Vokabular und den systematischen Vorstellungen, die auf der einen Seite Nutzer*innen und auf der anderen Seite Entwickler*innen von den Algorithmen haben. Es ist nicht nur so, dass damit beide Wissensbestände als gleichwertig betrachtet werden, sondern die Vignettenbildung lässt der Beschreibung der Nutzer*innen den Vorrang.

Technische Informationen ›plündern‹

Die Vignetten liefern die Grundlage für den dritten Schritt der Analyse. In meiner Feldforschung begann diese Arbeit mit einem Interview, bei dem ich einem Entwickler für Navigationssoftware erste Vignetten von Nutzerinteraktionen vorlegte. Auf das initiale Interview folgte ein Prozess, den Seaver als »Plünderung« (Seaver 2017: 7) oder Christin als »Tanz mit dem Algorithmus« (Christin 2020: 16) bezeichnet: An allen mir zugänglichen Stellen habe ich technische Informationen zum Routenplaner gesucht, informelle Gespräche mit Entwicklern geführt, technische Aspekte in der Informatik-Literatur recherchiert, Zeitungsartikel und Pressemitteilungen von Softwareunternehmen verfolgt, technische Vorträge, Präsentationen und Ausstellungen besucht. Stets ging es darum, noch genauer zu verstehen, wodurch die geschilderten Interaktionen technisch bedingt sind und warum sie so verliefen, wie sie die Fahrer*innen schilderten. Es ging darum zu verstehen, wie bestimmte Mechanismen und algorithmische Logiken technisch funktionieren »könnten«. Mit diesem Konjunktiv beziehe ich mich auf Wendy Chun, die zu bedenken gibt, dass es schier unmöglich ist, genaue Wahrheiten über die Funktionen einer Software zu finden. Auch Entwickler*innen können nur näherungsweise erschließen, wie unterschiedliche Softwarekomponenten im Zusammenspiel wirken (Chun 2011: 54). Mit ihrer Nähe zur Informatik bieten besonders die Ansätze aus den *Software Studies* die Möglichkeit, die Wirkung von Software hinsichtlich verschiedener Komponenten und Logiken technisch zu differenzieren.

Auch die Arbeitspraktiken von Entwickler*innen sind dabei relevant, da ihre alltäglichen Abwägungen und Kompromisse sich ebenso wie mathematische oder mechanische Logiken in die schlussendliche Routenberechnung wesentlich einschreiben. Eine solche Abwägung zeigt sich beispielhaft am Graphen, der als mathematisches Modell jene Straßen abbildet, die in die Routenberechnung eingehen (siehe Abbildung 2): Straßen werden darin zueinander ins Verhältnis gesetzt, indem Eigenschaften wie Länge und Geschwindigkeitsbegrenzungen einer Straße in »Kostenwerte« übersetzt werden. Um das Straßengeschehen für den Algorithmus im Graphen zu modellieren, müssen Software-Entwickler*innen zum Teil bestimmte Ausschlüsse vorwegnehmen, um die Berechnungszeit einer Route

möglichst kurz zu halten. Sogenannte Straßenfunktionsklassen[6] bieten dazu vorgefertigte Hierarchien, mit denen etwa verkehrsberuhigte Zonen kategorisch aus der Berechnung ausgeschlossen werden können, um eine Berechnungszeit zu beschleunigen. Solche Ausschlüsse entfernen gegebenenfalls auch Straßenoptionen, die potenziell genauso schnell zum Ziel führen. Doch Alltagskompromisse wie diese sind zwangsläufig in die Routenberechnung eingeschrieben, damit das Finden einer geeigneten Route überhaupt algorithmisch bzw. durch Computer berechnet gelöst werden kann.

Ein weiteres Beispiel ist die Erstellung des »Kostenwerts« einer Straße. Das Format des Kostenwerts gibt strenge Standards vor, welche Qualitäten einer Straße überhaupt in einer algorithmischen Routenberechnung berücksichtigt werden können und welche nicht. Das Format begünstigt selbstverständlich Eigenschaften, die sich leichter quantifizieren lassen als andere. Für Qualitäten wie die Schönheit oder Bequemlichkeit einer Route etwa bedarf es zahlreicher Umarbeitungen (Quercia 2014). Auch der Aufwand zur Erstellung des Kostenwerts für Straßen ist darum bestimmten Pragmatismen und ökonomischen Entscheidungen unterworfen. Entwickler*innen treffen zahlreiche solcher technischen Kompromisse, Pragmatismen und Abwägungen, die für das Routenergebnis bestimmend sind. Eine algorithmisch empfohlene Route im Straßenverkehr ist damit nicht objektiv kurz oder schnell, sondern wird von Entwickler*innen zu einem algorithmisch lösbaren Problem umgearbeitet. Schon allein solche technischen Konstruktionsentscheidungen haben soziale Relevanz für die Nutzer*innen und bleiben ihnen dennoch hinter der Anzeige eines eindeutig erscheinenden, algorithmischen Ergebnisses verborgen.

Die vermeintlich einfache Frage von Seiten der Nutzer*innen – Wie kommt die Software zu ihrem Ergebnis? – erlaubt es, hinsichtlich der am Ergebnis beteiligten Softwarekomponenten und Entwicklerpraktiken beliebig komplex zu werden. Aber der Ausgangspunkt an einer konkreten Nutzerfrage nimmt die wesentliche Setzung vor, dass der zunächst eingeschränkte Blick des Software-Nutzers auf das algorithmische Ergebnis eine erste Auswahl vornimmt. Es wird sichtbar, was Nutzer*innen von den zugrundeliegenden Logiken der Software sehen bzw. wissen können und was sie ebenso nicht sehen bzw. nicht wissen können. Ihre Plausibilisierungsstrategien zeigen, dass auch die Nutzer*innen stets mit Vorwissen in die

6 Die Straßenfunktionsklassen unterscheiden sich je nach Anbieter hinsichtlich Anzahl und Qualität ihrer Straßenklassen. Während TomTom neun verschiedene Funktionsklassen (Value FRC 0-8) anbietet, werden bei HERE fünf Funktionsklassen (Value FRC 1-5) unterschieden (siehe: https://developer.here.com/documentation/traffic/topics/resource-type-functional-class.html, zuletzt abgerufen am 23. Mai 2020. Für die Funktionsklassen von Open Street Map siehe https://wiki.openstreetmap.org/wiki/Key:highway, zuletzt abgerufen am 19. Februar 2018.

Interaktion mit Algorithmen gehen und ein Algorithmus auch für sie keine grundsätzliche Black-Box ist. Mit ihrem Blick können jene opaken Stellen in der Software ausfindig gemacht werden, die ein Problem bzw. eine Frage für die Nutzer*innen darstellen. Da Software-Entwickler*innen vornehmlich zu jenen technischen Prinzipien befragt werden, die die Nutzer*innen ausgewählt haben, bestimmen damit die Nutzer*innen die neuartigen Blickfelder auf die zunächst undurchsichtigen Schattierungen ihrer Alltagsalgorithmen.

Algorithmische Sichtbarkeiten mit architektonischen Metaphern erschließen

Algorithmische Blickfelder für neue Erklärbarkeiten kuratieren

Die konsequente Setzung, algorithmische Entscheidungssysteme mittels der Plausibilisierungsstrategien interessierter Nutzer*innen zu erforschen, hat den Vorteil, nicht nur die Wirkungen von Softwarefunktionen, sondern auch die Wirkung ihrer Opazitäten erschließen zu können. Schon die Opazität von Software erzeugt Ungleichheitsgefüge in der Präsentation von Zusammenhängen, noch bevor Software überhaupt ihre funktionale Wirkung entfaltet. Die Nutzerperspektiven verdeutlichen, wie Algorithmen Sichtbarkeiten strukturieren – auf sich selbst und ihre Bewertungskriterien, aber vor allem auf die sozialen Situationen, die sie mitbestimmen. Algorithmen regeln, welche größeren Bezüge und Zusammenhänge von Nutzer*innen nachvollzogen, verstanden und hinterfragt werden können. In ihnen materialisiert sich eine bestimmte Politik des Arrangierens von Blickfeldern, was aus welcher Perspektive gesehen, erkannt und gewusst werden kann: Software-Nutzer*innen sehen technische Funktionslogiken samt ihrer organisatorischen Hintergründe, mit denen sie interagieren, nur durch ein »kleines Schlüsselloch«, wie es Lucy Suchman formuliert (Suchman 2007: 11). Dass Algorithmen für Nutzer*innen opak sind, liegt deshalb nicht am Mangel von Nutzerkompetenzen, sondern es ist eine bauliche Beschaffenheit heutiger Alltagsalgorithmen, die zu diesen Opazitäten führt. Um bei räumlichen Metaphern zu bleiben, ›kuratieren‹ Algorithmen ähnlich wie Fenster und Wände in räumlichen Architekturen, ob, was, wann und wie Nutzer*innen eine bestimmte Information ›zu Gesicht‹ bekommen. Ähnlich zu anderen architektonischen Formationen üben auch Software-Arrangements regulatorische Kräfte auf ihre Nutzer*innen aus. Von Routenplanern unterstützte Fahrer*innen bewegen sich mit algorithmisch definierten Blickfeldern durch den städtischen Raum. Mit solchen räumlichen Metaphoriken lassen sich die von Software eingeführten Regelungen bestimmter Wissensbestände plastisch vor Augen führen.

Die Medienwissenschaftlerin Taina Bucher etwa beschreibt mit raumtheoretischen Analysekategorien die Wirkung von Alltagsalgorithmen am Beispiel des News-Feeds von Facebook (Bucher 2012). Bucher bezieht sich auf Michel Foucaults panoptische Architekturen und überträgt sein Konzept auf algorithmische Anordnungen: »An architectural perspective usefully highlights the ways in which spaces are ›designed to make things seeable, and seeable in a specific‹ way.« (Bucher 2012: 27) Insbesondere John Rajchmans Auslegung einer Foucault'schen Architektur zur Regelung von Sichtbarkeiten ist hierfür hilfreich: »Architecture helps ›visualize‹ power in other ways than simply manifesting it. It is not simply a matter of what a building shows ›symbolically‹ or ›semiotically‹, but also of what it makes visible about us and within us.« (Rajchman 1988: 103) Auch in der Programmierung, im Marketing und im Interface-Design von Software werden Blickfelder für die Nutzer*innen entschieden. Eine architektonische Perspektivierung verdeutlicht, dass das algorithmische Auswählen von Informationen technisch und medial notwendig und damit niemals neutral ist.

Für eine nutzer-zentrierte Perspektive gibt deshalb auch die aktuelle Stadtforschung Anregungen – im Konzept des dänischen Stadtplaners Jan Gehl zur »human-scale Architektur« (Gehl 2011). Als Gegenentwurf zur autofreundlichen Stadt nimmt Gehl bereits im Planungsprozess die Augenhöhe eines Menschen mit seinen Interessen, Empfindungen und Wünschen an die eigene Stadt als Lebensraum zum Ausgangspunkt. Architektonische Elemente wie Gebäude, Straßenverläufe und Plätze werden in Referenz und Beziehung zu Fußgängern gedacht und geplant. Bereits im gestalterischen Prozess wird die Perspektive und das Erleben eines Fußgängers zum Ausgangspunkt, der seine Stadt über bestimmte, klar definierbare Blickfelder wahrnimmt, die etwa durch seine Augenhöhe und seine Schrittgeschwindigkeit von fünf km/h definiert sind. Diese Perspektivverschiebung verändert die Planung einer Stadt und erzeugt andere Städte »at human scale« (Gehl 2011).

Auch für erklärbare Algorithmen gilt es, die Blickfelder von Nutzer*innen in einem ähnlichen Sinne neu zu denken. Alltagsalgorithmen basieren noch heute auf einem Gestaltungsprinzip, das für das Interface-Design von Computern im Kalifornien der 1970er Jahre entwickelt wurde. Damals sollten Nutzer*innen von Computern nicht länger über textbasierte Kommandozeilen zwischen den Programmen navigieren, sondern stattdessen über Fenster, Icons, Menus und Cursor. Damit änderte sich der Dialog zwischen Nutzer*innen und Computern (Bunz 2019: 76). Die damalige Entwicklung des *graphical user interface* (GUI) am Xerox PARC trug wesentlich dazu bei, dass sich Personal Computer als Massenprodukt am Markt etablieren konnten (Chun 2011: 59). Mit diesem Wandel im Technikdesign verschwanden die technischen Prozesse zunehmend aus dem Blickfeld der Nutzer*innen.

In umgekehrter Weise kann heute für die Erklärbarkeit von Algorithmen wiederum gefragt werden, welche Perspektiven auf technische Abläufe, Programme

und Kommandos für eine emanzipative Techniknutzung für die Nutzer*innen nötig sind, um algorithmische Empfehlungen prüfen, persönliche Daten verwalten und schützen, fremde Zugriffe kontrollieren oder eine Sachverständigkeit darüber entwickeln zu können, welche Kriterien für die algorithmische Berechnung eine Rolle spielen. Ein ›Kuratieren‹ neuer Blickfelder auf einzelne Softwarelogiken könnte wesentliche Informationen für die Nutzer*innen offenlegen.

Nutzer*innen an der Gestaltung von Erklärbarkeiten beteiligen

Damals wie heute ermöglicht die Einbeziehung der Nutzer*innen einen solchen Perspektivwechsel. Denn solange in der Technikgestaltung zu wenige dieser Erklärung schaffenden Blickfelder für die Nutzer*innen zur emanzipativen Techniknutzung angelegt sind, können bauliche Software-Intransparenzen nur über die Plausibilisierungsstrategien der Nutzer*innen kompensiert werden. Diese Strategien machen sichtbar, ob es die Logik eines Algorithmus oder selbst erzeugte Bewegungsdaten oder eine gezielte Designentscheidung eines Managers oder einer Software-Entwicklerin ist, die einen entscheidenden Unterschied in der Routenberechnung machen. Nutzerpraktiken dienen zum Aufschlüsseln und Auswählen operativer, opaker Softwarelogiken und -prinzipien, die einer sozialwissenschaftlichen Analyse und der Plausibilisierung überhaupt bedürfen. Konzipiert man bei der Analyse ›künstlich-intelligenter‹ Systeme auch die Nutzer*innen als ›intelligent‹, so lassen sich zum einen die Nutzerbilder aktueller Softwaregestaltungen kritisch prüfen. Zum anderen kann ein solcher praxistheoretischer Ansatz auch zur Gestaltung technischer Systeme anregen, die aktiv partizipierende Nutzer*innen mitdenkt.

Bereits seit den 1970er Jahren setzt sich das *Participatory Design* dafür ein, Nutzer*innen von Technologien in die Designprozesse von Technologien miteinzubeziehen (Greenbaum et al. 1991; Simonson et al. 2013). Bereits damals war durch die Entwicklung neuer Computertechnologien die Sorge groß, neue Technologien könnten zur Dequalifizierung von Arbeitskräften beitragen. Im technik-intensiven Arbeitskontext bildeten sich darum Teams aus Angestellten und Gewerkschaften mit dem Ziel, Computer mit Rücksicht auf eine demokratische Teilhabe am Arbeitsplatz so zu gestalten, dass sie Arbeiter*innen und Unternehmer*innen gleichermaßen dienen. Technologien sollten es Arbeiter*innen ermöglichen, eigenes Wissen und lokale Kompetenzen mit automatisierten Prozessen zusammenzubringen. Im *Participatory Design* wurden deshalb methodische Ansätze entwickelt, Technologien nicht nur für, sondern mit den Nutzer*innen zu gestalten. Diese Forschungstradition unterstützt bis heute demokratisch förderliche Innovationskonzepte und Technologien (Ehn et al. 2014: 7), doch Vicki S. Napper weist auch darauf hin, dass diese Art der Technikgestaltung im politischen Zeitgeist der 1980er Jahre eingebettet war:

> »The European labor market fosters this type of alliance through strong labor unions demanding a say in the worker's environment. Although the world marketplace does not always provide employees with the final approval, the idea of ›reflecting on the practices‹ and approaching the computer system design with cooperative involvement of all parties is a timely idea.« (Napper 1994: 97)

Auch Pelle Ehn bemerkt, dass die Netzwerke demokratischer Technikentwicklung in den 2000er Jahren durch das Aufkommen des Neoliberalismus als neues Paradigma vehement gestoppt wurden (Ehn 2020). In Deutschland lag in den 1970er Jahren ein besonderer Fokus auf partizipativer und demokratischer Arbeitsplatz- und Technikgestaltung, wie Ernst A. Hartmann am Beispiel des Forschungsprogramms »Humanisierung des Arbeitslebens« der damaligen sozialliberalen Koalitionsregierung darlegt (Hartmann 2015: 10f.). Angesichts der aktuellen Forderung nach Erklärbarkeit von Algorithmen sowie nach gesellschaftlicher Digitalkompetenz können diese wertvollen Ansätze und Partizipationsstrategien für heutige Technikangebote neu interpretiert werden. Nur weil Algorithmen erklärt werden sollen, muss die Technologie keinesfalls weniger komplex werden. Der dargestellte Forschungsansatz zeigt, dass es genügen könnte, Opazitäten situativ an einzelnen Stellen für die Nutzer*innen aufzudecken.

Demokratisches Algorithmenwissen

Das hier vorgeschlagene Forschungsdesign des dreischrittigen Abgleichs zwischen der Sammlung von Nutzereindrücken, der Beschreibung opaker Blickfelder in Mensch-Maschine-Interaktionen mittels Vignetten und der anschließenden Recherche einzelner technischer Informationen rückt Softwarekomponenten in den Fokus, die zwar algorithmisch berechnete Ergebnisse wie die Wahl einer Route prägen, aber für die Nutzer*innen opak sind. Technisch formale Bedingungen wie Algorithmen, Modelle, Graphen und Kostenwerte können so für eine nutzerzentrierte Erklärbarkeit aufbereitet werden. Im Folgenden werden drei Anwendungsgebiete für algorithmische Erklärungen skizziert:

Erstens etablieren modellhafte Analysen von Alltagsalgorithmen wie dem A*-Algorithmus nutzerzentrierte Vorstellungen, was ein Algorithmus leisten kann und was nicht. Dies trägt zur Entwicklung gesellschaftlicher Digitalkompetenzen bei, die wiederum informiertere demokratische Debatten zu Algorithmen samt ihrer Möglichkeiten und Grenzen motivieren können. Eine gesellschaftliche Grundkenntnis über Alltagsalgorithmen kann zudem für die Diskussionswürdigkeit verschiedener Algorithmeneinsätze sensibilisieren: Das skizzierte Beispiel des Ausschlusses verkehrsberuhigter Straßen aus der Routenberechnung zeigt als recht unproblematisches Anwendungsszenario, inwiefern Entwickler*innen

schon allein technisch bedingt – also ohne weitere Absichten und Interessen – Kategorisierungen und Hierarchisierungen für das Funktionieren von Algorithmen vornehmen müssen, die soziale Ungleichheiten gegebenenfalls unterstreichen bzw. verstärken.

Zweitens: Bedienoberflächen von Alltagsalgorithmen bieten nur begrenzt Raum, um Nutzer*innen technische Erklärungen in der jeweiligen Entscheidungssituation zu geben. Dies zeigt sich am schlichten, reduzierten Interfacedesign vieler Richtungspfeil-Ansichten von Routenplanern. Nutzerzentrierte Methodiken zur Erklärbarkeit von Algorithmen können neue Impulse für Gestaltungsprinzipien setzen, die beispielsweise näher an den technischen Grundlogiken bleiben, wie dem Graphen als Repräsentation des Straßennetzes (Abbildung 2). Grenzen, die algorithmische Modellierungen zwangsläufig haben, sowie Kostenwerte und Gewichtungen könnten darüber von den Nutzer*innen nachvollzogen werden. Ihr eigenes Kontextwissen ließe sich besser abgleichen und in algorithmisch gestaltete Entscheidungsprozesse miteinbinden. Künstlich-intelligente Algorithmen könnten sich konstruktiv mit intelligenten Nutzungsweisen ergänzen.

Drittens: Bereits bei der Programmierung von Algorithmen kann die nutzerzentrierte Erklärbarkeit als Gestaltungsprinzip einbezogen werden. Technische Kompromisse wie der Ausschluss verkehrsberuhigter Zonen zugunsten schnellerer Routenberechnungen müssen dann bereits im Softwarebüro diskutierbar werden. Madeleine Akrichs Konzept der *i-methodology* besagt, dass Entwickler*innen eine Software häufig so implementieren, wie sie ihnen ganz persönlich wünschenswert oder sinnvoll erscheint (Akrich 1995: X). Das Problem besteht weniger darin, dass Entwickler*innen in dieser Weise vorgehen, sondern dass selbige Methode nicht an die Nutzer*innen kommuniziert wird. Stattdessen erscheint eine vermeintlich – weil nach mathematischen Bedingungen optimal – ›berechnete‹ Route auf dem Display. Auch für Entwickler*innen bedarf es darum neuer Strategien zur Sensibilisierung, welche ihrer situativen Annahmen und alltäglichen Problembehebungen einen konkreten Unterschied für das Berechnungsergebnis machen. Tests, Tandem-Programmierung und Code-Dokumentationen schaffen erste Möglichkeiten zur Erklärbarkeit von Code im Entwickleralltag. Doch die Einbindung ethnografischer Ansätze kann diese Ansätze produktiv um die nutzerzentrierte Perspektive ergänzen. Die Interpretationen von Algorithmen blieben damit nicht länger im Vokabular der Entwickler*innen, sondern würden auch von den Nutzer*innen ausgedacht und formuliert.

Je komplexer und undurchsichtiger KI-basierte Entscheidungssysteme sind, desto weniger können Nutzer*innen mit eigenen Kenntnissen an der Entscheidung partizipieren, selbst wenn sie über Kontextwissen zur jeweiligen Situation verfügen. Je nach Publikum müssen die Blickfelder auf einzelne technische Logiken spezifiziert werden. Die hier formulierten Ansätze erfordern weitere Debatten darüber, wie nutzerzentrierte Erklärungen gestaltet werden können: Wie sieht die

Benutzeroberfläche eines Routenplaners aus, die Blickfelder auf Software-Logiken freigibt – den rechnenden Algorithmus, die Graphenkanten oder verwendete Parameter und Kostenfunktionen – ohne das Display zu überfrachten? Am Beispiel des Routenplaners werden mehr Ansatzpunkte offenkundig als nur die Veränderung von Benutzerschnittstellen. Vielmehr stellt sich die Frage, was auf gesellschaftlicher Ebene oder im Entwicklerbüro gewusst oder erlernt werden muss, um vielfältige Legitimierungsprozesse von Algorithmen anzuregen. In einer demokratischen Gesellschaft genießt die Möglichkeit zum Aushandeln n einen hohen Wert und Algorithmen zählen als vielgenutzte Entscheidungshilfen zunehmend zu aushandlungswürdigen Aspekten des Alltags.

Literatur

Akrich, M. (1995). User Representations: Practices, Methods and Sociology. In *Managing Technology in Society. The Approach of Constructive Technology Assessment,* eds. Rip, A., Schot, J. and Misa, T. J., 167-184. London: Pinter Publisher.

Barocas, S., Rosenblat, A., boyd, d., Gangdharan, S.P., and Yu, P. (2014). *Data & Civil Rights: Technology Primer*. Data & Civil Rights Conference, October 2014. https://papers.ssrn.com/sol3papers.cfm?abstract_id=2536579. Zugegriffen: 28. August 2020.

Bucher, T. (2012). *Programmed Sociality: A Software Studies Perspective on Social Networking Sites*. Oslo: Universität Oslo.

Bunz, M. (2019). The Force of Communication. In *Communication*, eds. Bialski, P., Brunton, F. and Bunz, M., 51-92. Lüneburg: meson press.

Burrell, J. (2016). How the Machine ›thinks‹. Understanding Opacity in Machine Learning Algorithms«. *Big Data & Society* 3(1): 1-12.

Christin, A. (2020). The ethnographer and the algorithm: beyond the blackbox. *Theory and Society* 49: 897-918.

Chun, W. (2011). *Programmed Visions: Software and Memory, Software Studies*. Cambridge, MA: MIT Press.

Ehn, P., E. M. Nilsson und R. Topgaard, eds. (2014). *Making Futures: Marginal Notes on Innovation, Design, and Democracy*. Cambridge, MA: MIT Press.

Ehn, Pelle (2010). Exploring the Scandinavian Participatory Design Tradition, Medea Talks #17, School of Arts and Communication, Malmö, https : https://www.youtube.com/watch?v=g4xofWbhVu8. Zugegriffen: 30. August 2020.

Gehl, J. (2011). *Life between Buildings: Using Public Space*. Washington, DC: Island Press.

Greenbaum, J. and M. Kyng, eds. (1991). *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, N.J: Erlbaum.

Hartmann, E. A. (2015). Arbeitsgestaltung für Industrie 4.0: Alte Wahrheiten, neue Herausforderungen. In *Zukunft der Arbeit in Industrie 4.0*, Hg. Botthof, v. A. und Hartmann, E. A., 9-22. Berlin: Springer Vieweg.

Knorr Cetina, K. (1999). *Epistemic cultures: How the sciences make knowledge*. Cambridge: Harvard University Press.

Latour, B. (2010). *The making of the law: An ethnography of the Conseil d'Etat*. London: Polity.

Napper, Vicki S. (1994). A Review: Design at Work: Cooperative Design of Computer Systems. I *Educational Technology Research and Development* 42(1): 97-99.

Pariser, E. (2012). *The Filter Bubble: What the Internet Is Hiding from You*. London: Penguin Books.

Pentenrieder, A. (2020). *Algorithmen im Alltag. Eine praxistheoretische Studie zum informierten Umgang mit Routenplanern.* Frankfurt: Campus.

Quercia, D., Schifanella, R. and Aiello, L. M. (2014). The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City. *ACM Press*. 116-125.

Rajchman, J. (1988). Foucault's Art of Seeing. *October* 44: 88-117.

Rosenblat, A. (2018). U*berland: How Algorithms Are Rewriting the Rules of Work*. Oakland, CA: University of California Press.

Seaver, N. (2017). Algorithms as Culture: Some Tactics for the Ethnography of Algorithmic Systems. *Big Data & Society* 4(2): 1-12.

Seaver, N. (2019). »Interpretability,« or Learning to Listen to Algorithms, Colloquium Talk Lausanne, unveröffentlichtes Manuskript.

Seyfert, R. und Roberge, J. (2017). *Algorithmuskulturen. Über die rechnerische Konstruktion der Wirklichkeit*. Bielefeld: transcript.

Simonsen, J. and Robertson, T., eds. (2013). *Routledge International Handbook of Participatory Design*. New York: Routledge.

Sonnenburg, S. (2019). How TomTom is using AI to Create World-Class Maps. Vortrag: idalab seminar #21, Berlin.

Spielkamp, M. (2019). *Automating Society. Taking Stock of Automated Decision-Making in the EU.* Berlin: algorithm watch.

Star, S. L. (2017). *Grenzobjekte und Medienforschung*, Hg. S. Gießmann und N. Taha. Bielefeld: transcript.

Strübing, J. (2002). Just Do It? Zum Konzept der Herstellung und Sicherung von Qualität in Grounded Theory-basierten Forschungsarbeiten. *Kölner Zeitschrift für Soziologie* 54(2): 318-342.

Suchman, L. (2007). *Human-Machine Reconfigurations: Plans and Situated Actions.* 2nd rev. edition. Cambridge: Cambridge University Press.

Töpfer, Klaus (2016). Unterwegs als Kurierfahrer bei Foodora. Radeln against the Machine. In: taz. *die tageszeitung*, Öko/Arbeit, url: www.taz.de/!5292438/. Zugegriffen: 09. Dezember 2018.

Velden, L. (2014). Der A*-Algorithmus. url: https://www-m9.ma.tum.de/graphalgorithms/spp-a-star/index_de.html. Zugegriffen: 09. Dezember 2018.

Wachter, S., Mittelstadt, B. and Russell, C. (2017). Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*.

Weizenbaum, J. (2008). *Die Macht der Computer und die Ohnmacht der Vernunft*. Nachdruck. Frankfurt a.M.: Suhrkamp.