Media in Transition 8, Cambridge, MA, April 2013

Knowing algorithms

Revised, February 2014

Nick Seaver
Department of Anthropology, UC Irvine
Intel Science and Technology Center for Social Computing

Unpublished draft. If you find this useful or interesting, please let me know: nseaver@uci.edu

I. Definition

> *What are algorithms? Why is the study of algorithms worthwhile? What is the role of algorithms relative to other technologies used in computers?*
>
> *Informally, an* **algorithm** *is any well-defined computational procedure that takes some value, or set of values, as* **input** *and produces some value, or set of values, as* **output**. *An algorithm is thus a sequence of computational steps that transform the input into the output.*
>
> *We should consider algorithms, like computer hardware, as a* **technology.**
> (Cormen, et al. 2009:5, 13)

Algorithms have become objects of concern for people outside of computer science. As more and more of human life is conducted alongside and through computers, contoured by algorithmic selection both online and off, people who once had little interest in the workings of computers have growing concern about their effects. This concern has manifested in an explosion of popular and academic productions: books, articles, op-eds, blog posts, conferences, and research programs engage with algorithms in terms that never appear in the *Introduction to Algorithms* textbook from which I took the epigraph above.

For undergraduates in computer science, Cormen et al.'s textbook is the definitive source for knowledge about algorithms. It is a massive book: 1,000+ pages containing descriptions of basic algorithms and the data structures on which they operate. It teaches how to design an algorithm, to determine its efficiency, and to prove its optimality. For Cormen et al., knowledge about algorithms is a matter of mathematical proof, and "analyzing an algorithm has come to mean predicting the resources that the algorithm requires" (2009:21). Other knowledge about algorithms — such as their

applications, effects, and circulation — is strictly out of frame here, save for the obligatory mention in the introduction of genomics, network routing, cryptography, and resource allocation. When algorithms move from academia to commercial application, they retain many of the qualities granted to them by Cormen et al.'s *Introduction.* Algorithms *per se* are supposed to be strictly rational concerns, marrying the certainties of mathematics with the objectivity of technology.

Those of us on the outside have different interests. The "us" here (and through the rest of this paper) refers to a set of social scientists and humanists interested in how computation is interacting with, changing, and being changed by our disciplines' traditional objects: culture, society, the public sphere, interpretation, politics, and so on.[1] For now, I am speaking from and to this group of social scientists and humanists, although I suspect that some of the points I raise here apply beyond us — even to the engineers and scientists we may think of as "insiders" to algorithmic processes. Indeed, part of my goal here is to undermine some common assumptions about how knowledge and criticism regarding algorithms is distributed across the insider/outsider divide.

Our interest in algorithms is primarily shaped by a concern about what happens when the rigid, quantitative logic of computation tangles with the fuzzy, qualitative logics of human life. Like algorithms, this concern is not new: we have known similar questions under the names "rationalization" or even "modernization" before. The encounter between culture and technology seems to invite grandiose claims about intrinsic "cultural logics" of broad and heterogeneous technical formations, even as a significant and growing body of scholarship in STS and affiliated fields undercuts the assumed opposition between technology and culture, emphasizes the emergent properties of technical systems in use, and points to the importance of historical and contextual specificity in understanding sociotechnical arrangements.

Researchers interested in the empirical specificities of algorithms in action (rather than broad, polemic statements about computation and culture) have a problem: How do we know about algorithms? In this paper I will argue that, in spite of the rational straightforwardness granted to them by critics and advocates, "algorithms" are tricky objects to know. This is not a particularly novel position: algorithms are commonly considered inscrutable as a result of 1) the proprietary nature of commercial systems, and 2) the technical know-how required to make sense of them. But what makes knowing algorithms *really* tricky, I propose, is that these two issues are only superficial

[1] To save commas, and because I am a cultural anthropologist, I will use "culture" through the rest of this paper to stand for this humanistic hodgepodge, following to an extent E.B. Tylor's foundational omnibus definition of culture from 1871: "that complex whole which includes knowledge, belief, art, morals, law, customs, and any other capabilities and habits acquired by man as a member of society".

problems: they suggest that the barriers to knowledge about algorithms are access and expertise. If only we had access and if only we had the expertise, then all would be clear and we could get on with our critical assessments. I want to suggest that in fixating on access and expertise, we reify a deficient understanding of how algorithms exist and work "in the wild," outside of the theoretical definitions found in computer science textbooks. In the rest of this paper, I will outline what I see as the dominant critical approach to knowing algorithms, describe some problems that face this approach, and suggest an alternative definition of "algorithms" as they concern those of us on the outside.

II. ***Anagnorisis***

In a TED Talk based on his book, *The Filter Bubble* (2011a), internet activist and entrepreneur Eli Pariser describes a moment of recognition — the realization that his experience online was being quietly mediated by algorithmic filters:

> I first noticed this in a place I spend a lot of time: my Facebook page. I'm progressive politically, but I've always gone out of my way to meet conservatives. I like hearing what they're thinking about; I like seeing what they're linking to; I like learning a thing or two. So I was kind of surprised, when I noticed one day that the conservatives had disappeared from my Facebook feed. And, what had turned out was going on was that Facebook was looking at which links I clicked on, and it was noticing that actually I was clicking more on my liberal friends' links than on my conservative friends' links. And without consulting me about it, it had edited them out. They disappeared. (2011b)

Pariser's story has a few remarkable features. First, it is almost certainly a fiction. As director of MoveOn.org, a major online liberal activist organization, Pariser knew his way around the internet, and it is unlikely that he was surprised that algorithmic filtering existed on a service like Facebook. His story is better understood as a kind of parable, and as a parable it bears its second interesting feature: the entanglement of concerns about hiddenness and morality. Pariser poses Facebook's filtering as a moral problem in two ways: it hides divergent viewpoints, inhibiting debate and the establishment of a public sphere, and it hides itself, preventing users from realizing what has happened. The solution is transparency: filters and the content they hide should be made visible.

Pariser's moment of recognition is a didactic model for knowledge about algorithms. Where filters obscure, recognition brings sudden clarity. Such moments are common features of popular critical writing about algorithms. They are narrative devices in a

literal sense: *anagnorisis* is Aristotle's term from the *Poetics,* a "change from ignorance to knowledge, producing love or hate" (1:XI). When Oedipus discovers his true parentage or King Lear discovers the treachery of his daughters, that is *anagnorisis*. *Anagnorisis* presupposes a particular definition of knowledge: the dramatic realization of a preexisting fact (with the drama often heightened by the fact that the audience has known it all along). By focusing attention narrowly on the moment of discovery, *anagnorisis* tempts us to take the facts discovered for granted — in the fictional worlds of Oedipus and Lear, there is no doubt of Jocasta's incest or Goneril and Regan's treachery.[2] Once recognized, knowledge is no longer in question.

In this model, knowing algorithms is a matter of revealing them. Once the basic facts of their functioning are revealed, then they can be critiqued. To know an algorithm is to get it out in the open where it can be seen. However, work in science studies, particularly its constructivist strains, has taught us to be suspicious of overly simple stories of revelation. These stories, we have learned, often elide important details about how knowledge is achieved and produced. Constructivist accounts of knowledge production emphasize the processes through which knowledge is achieved, not as the overcoming of barriers or pulling back of veils to reveal what is really going on, but as interactional work that produces local and contingent truths.

This may sound like a strange attitude to take towards algorithms, which are supposed to be human-designed paradigms of rationality and objectivity. Surely algorithms cannot be mysterious and hard to specify in the same way that solar neutrinos or deep-sea thermophilic microbes are, right? Taking my lead from science studies, I want to suggest that there is more going on in the making of knowledge about algorithms than the revelation of plain fact. What we recognize or "discover" when critically approaching algorithms from the outside is often partial, temporary, and contingent.

III. Experimentation

Faced with the corporate secrecy of a company like Facebook, some critical researchers turn to experimentation or reverse engineering to determine what is happening behind the curtain. In such studies, researchers augment their claims about algorithms with casual experiments conducted through the user interface: searching the same term through Google on multiple computers, systematically interacting with Facebook to see what content is surfaced, and so on. Within the usual constraints of social scientific and humanistic research, these experiments are typically run by one or two people, through ordinary user accounts. While these experiments can provide evidence that filtering has

[2] As these examples suggest, this way of thinking about knowledge has a sexual politics as well.

occurred (e.g. by receiving different search results from computers in different locations), they face serious limits when it comes to determining what algorithms are like with any specificity.

Personalization
Given that personalization algorithms by definition alter one's experience according to interactions with the system, it is very difficult, if not impossible, for a lone researcher to abandon the subject position of "user" and get an unfiltered perspective. In the case of many recommender systems, "everything is a recommendation" (Amatriain and Basilico 2012), and all interactions with the system are tailored to specific user accounts.

To provide an example from my own fieldwork, I recently interviewed a scientist who works for an online radio company. When I asked him about the playlisting algorithm that picks which song to play next, he corrected me: there is not one playlisting algorithm, but five, and depending on how a user interacts with the system, her radio station is assigned to one of the five master algorithms, each of which uses a different logic to select music. What does this mean for the researcher who assumes she is experimenting on one "algorithm"? When not only the results but the logic by which results are chosen are adapting to the individual user, according to invisible meta-logics, single user experimentation can't begin to cover all the possible permutations. Some researchers have attempted to overcome this problem through the use of software bots, which, posing as users, can cover more ground than a single human; while promising, such approaches may be limited by the ability of the services in question to "sniff out" non-human patterns of interaction.

A/B Testing
At a recent conference for designers of algorithmic recommender systems, Ron Kohavi, a research scientist working on Microsoft's Bing search engine, presented a keynote on "Online Controlled Experiments" (Kohavi 2012). These experiments, or "A/B tests," test modifications to Bing, from alternate designs of the "Search" button (should it have a magnifying glass or not?) to details of the ranking function (what are the best criteria for sorting the top five results?). At any moment, the engineers working on Bing are running about fifty of these experiments, sorting users into experimental groups and serving them with different versions of the site. In a given visit to Bing, a user will be part of around ten different experiments. Depending on how well these modifications perform according to various metrics (clicks, subsequent searches, etc.), they are incorporated into the system.

As a result of these experiments, there is no one "Bing" to speak of. At any moment, there are as many as *ten million* different permutations of Bing, varying from interface

design to all sorts of algorithmic detail. A/B testing is now standard for large web companies, and there are even services that algorithmically assess the tests' results, automatically implementing successful modifications on the fly. While we try to experiment on algorithms, they are experimenting on us. This raises a number of issues for would-be experimentalist critics: If we want to talk about how Bing works, what are we talking about? Features shared across the ten million Bings? One of ten million possible combinations? How do we know our status relative to the test groups? Rather than thinking of algorithms-in-the-wild as singular objects to be experimented upon, perhaps we should start thinking of them as populations to be sampled.

Change Over Time

David Karpf (2012) has written about the problems that face social scientists working in "internet time." The online landscape moves quickly, unlike social scientific research, publication timelines, or our traditional objects of study. What are we to make of claims about what "the internet" is like that are recent (in academic time), but pre-date the introduction of Twitter, the opening of Facebook to non-academic users, or the use of YouTube for US presidential debates? Many comparative claims are compromised by the fact that the internet has changed beneath them. If research relies on *ceteris paribus* assumptions — that while we examine some variable, all else is held stable — then the quick pace of online change poses serious issues.

While Karpf is concerned with the broader internet ecology, the *ceteris paribus* problem plagues even short term experimental engagements with algorithms online. Following from the A/B testing described above, algorithmic systems are constantly changing. These changes may be minor or major, but significantly, they will often be invisible to users. For would-be experimenters, this rapid evolution is a serious validity threat: You can't log into the same Facebook twice.

Experimental methods are well-suited to objects we have reason to believe are stable. While it makes sense to presume that gravity at the Earth's surface is more or less consistent, we have no such reason to believe that about the algorithmic operations under the hood of major search engines or social media services. Perhaps the persistence of experimentation belies a desire on the part of researchers: *if only* our objects of interest had a stable underlying logic, our jobs would be much easier.

IV. Transparency

The knowledge problems faced by experimental approaches are fairly straightforward. As outsiders, "we" do not know for sure what is happening behind the curtain, but as I've described, we have reason to believe that it is constantly changing and, even at a single moment, multiple. If we were engineers at a high enough level inside Facebook, these problems in their most obvious form would go away: we could know what test group a user had been assigned to, we could know what changes had been made to the architecture, and depending on how much authority we had, we might even be able to hold algorithmic variables constant to achieve *ceteris paribus* conditions. In short, the problem for experimentation is that we are not insiders — our experiments are designed to get at knowledge that engineers already have and are confounded by hidden variables that engineers already know about and control.

Calls for transparency are aimed at resolving this situation — to let knowledge about how algorithms actually work out from behind the curtain. If these hidden details were better known, the argument goes, then we could engage in more effective critique and there would be incentive to design algorithms more justly. However, transparency is not as clear as it seems. As we know from anthropological research on governments, regulation, and "audit culture," transparency in practice is complicated and by no means a guarantee that the information provided will be complete, accurate, or understandable (Strathern 2000; Hetherington 2011; Hull 2012). Transparency has its limits (Ziewitz 2013). For the developers of algorithmic filtering systems themselves, there is a concern that revealing the details would render their algorithms useless, enabling bad-faith actors to game the system (not to mention aiding corporate competitors), thus negating transparency's benefits (Granka 2010).

At their most simple, calls for transparency assume that someone already knows what we want to know, and they just need to share their knowledge. If we are concerned about Google's ranking algorithm for its search results, presumably that knowledge exists inside of Google. The moral rhetoric accompanying calls for transparency assumes that Google does not or cannot have critical perspective, so its inside knowledge should be passed along to those who do, or that transparency is a moral, democratic end in itself. While transparency may provide a useful starting point for interventions, it does not solve the problem of knowing algorithms, because not everything we want to know is already known by someone on the inside. Here, I want to point to two issues that pose problems for transparency: complexity and inclusion.

Complexity

In spite of what our formal definition might lead us to believe, algorithms in the wild are not simple objects. In a recent conference presentation, Mike Ananny described his encounter with an engineer working on Google's Android Marketplace, an online store for purchasing smartphone applications. Ananny had written an article about an offensive association made by the Android Marketplace's recommendation system (Ananny 2011): while looking at an application targeted at gay men, the recommender suggested an application for locating the residences of sex offenders. Several days after the article circulated online, he was contacted by an engineer at Google who offered to talk about the unfortunate association. While the "error" had quickly been manually corrected, it was still not clear why it had emerged in the first place. The operations of the recommender algorithm, incorporating machine learning techniques and changes made by numerous teams of numerous people, all accumulated over time, were so complex that there was no simple bit of code to point to as a cause.

The apparent simplicity and determinism of algorithms in theory does not account for their features in practice. In assuming that engineers have complete understanding of the systems they create, we make a mistake: first, these systems are works of collective authorship, made, maintained, and revised by many people with different goals at different times; second, once these systems reach a certain level of complexity, their outputs can be difficult to predict precisely, even for those with technical know-how. Simply making the system transparent does not resolve this basic knowledge problem, which afflicts even "insiders." A determined focus on revealing the operations of algorithms risks taking for granted that they operate clearly in the first place.

Inclusion

In presenting my own research on music recommendation algorithms, I am frequently asked questions about "inclusion": Does a given algorithm include lyrics when deciding that you might like a song? Does it include details about the music "itself" (tempo, key, etc.) or does it only include other listeners' ratings? The prevalence of these questions indicates a dominant attitude toward knowledge about algorithms: to know an algorithm is to know its decision-making criteria. Calls for transparency share this attitude: What is in there? We want to know!

However, the "what" of algorithmic inclusion is often less significant than the "how": to include something requires decisions about how to calculate and represent it. For example, the main variables included in Facebook's EdgeRank algorithm, which determines what content shows up on a user's news feed, are well publicized: it takes into account a user affinity score (a measure of affinity between the viewing user and the content posting user), a weight for different types of content (photos, for example,

are reportedly weighted higher than text alone), and a time-decay (older material is less likely to show up).[3] While this information is posted all over websites that claim to help brands promote themselves on Facebook, it tells us very little. How are user affinity scores calculated? How are content weights calculated? How do these calculations vary across different user types? None of these variables tell us more than we might have already guessed based on the fact that Facebook users share different kinds of material with each other over time. Add to this the fact that Facebook claims that EdgeRank contains other variables which are kept secret, and it becomes clear that even when we "know" the algorithm in broad strokes, we actually know precious little.

If we are interested in talking about algorithms' cultural effects, then it is not enough to know that something called "affinity" is included. Our questions should be more ambitious: What is affinity? How is it defined? What experiences do the engineers producing affinity scores draw on as they attempt to formalize it? How might these formalizations differ if we started from different assumptions and experiences? To assume that an algorithm can simply "include" something like affinity (or, for that matter, even such apparently objective features as tempo or key) is to miss the amount of interpretive, cultural work required to translate these features into computable form. It is to mistake maps for territories.

V. Redefinition: Toward the Ethnographic Study of Algorithmic Systems

The basic problem pointed to by these issues is this: the algorithms we are interested in are not really the same as the ones in *Introduction to Algorithms*. It is not the algorithm, narrowly defined, that has sociocultural effects, but *algorithmic systems* — intricate, dynamic arrangements of people and code. Outside of textbooks, "algorithms" are almost always "algorithmic systems."

We have inherited a number of technical distinctions from computer scientists for talking about algorithms: the difference between algorithms and data structures; the difference between an algorithm expressed theoretically and its implementation in a particular programming language; the difference between formally defined algorithms and ad hoc heuristics. These distinctions can be useful for building computational systems, and they are central elements of the emic understanding of computation among software engineers. However, they do not capture the dynamism and imprecision of algorithmic systems: we do not interact with algorithms and data structures in isolation from each other, there are no theoretical algorithms in actually

[3] Since the original drafting of this piece, it has come out that "EdgeRank is dead" (McGee 2013), replaced with a more complex blend of variables. Internet time strikes again.

existing software systems, and while a given algorithm may be well defined, the selection among algorithms may yet be ad hoc.

When we realize that we are not talking about algorithms in the technical sense, but rather algorithmic systems of which code *strictu sensu* is only a part, their defining features reverse: instead of formality, rigidity, and consistency, we find flux, revisability, and negotiation. The use of phrases like "the Google algorithm" or "the Facebook algorithm" should not fool us into thinking that our objects are simple, deterministic black boxes that need only to be opened. These algorithmic systems are not standalone little boxes, but massive, networked ones with hundreds of hands reaching into them, tweaking and tuning, swapping out parts and experimenting with new arrangements. If we care about the logic of these systems, we need to pay attention to more than the logic and control associated with singular algorithms. We need to examine the logic that guides the hands, picking certain algorithms rather than others, choosing particular representations of data, and translating ideas into code.[4]

This might be understood as a call to examine empirically the contexts — social, cultural, political, economic, legal, institutional, etc. — in which algorithms are developed, and that would be a welcome addition to the current state of algorithmic criticism. But a focus on context keeps algorithms themselves untouched, objective stones tossed about in a roily social stream. We should not abandon concern with technical details or "close readings" of algorithmic function; contextualization is not an explanatory panacea. My point is that when our object of interest is the algorithmic system, "cultural" details *are* technical details — the tendencies of an engineering team are as significant as the tendencies of a sorting algorithm. This is not so much an attempt to add the cultural back onto the technical as it is a refusal of the cultural/technical distinction as a ground for analysis.

This redefinition should be familiar to scholars of STS and the anthropology of technology, and it has significant consequences for how we examine the ethics of algorithms and their governability. Rather than starting our critiques from the premise that we already know what algorithms do in relation to culture — they reduce, quantize, sort, and control — we take that as our research question. How do algorithmic systems define and produce distinctions and relations between technology and culture?

---

[4] Of course, we do not always have the kind of access we might like to the systems we want to know about. In these cases we might have to resort to assessments of studies of "culture at a distance" or "reverse engineering" (Benedict 1946; Diakopoulos 2014), but as I have outlined here, these approaches face significant shortcomings that should not be ignored.

With this as our guiding question, the relationship between "outsider" humanistic or social scientific critics and "insider" engineers changes. When all there is to know about an algorithm is its function and effect, then expertise is neatly split: engineers know about functions, social scientists about consequences. But, if we're willing to muddy the waters a little, a space opens up for engineers and social scientists to critically engage with algorithmic systems together. Algorithmic systems, with their blend of "technical" and "cultural" concerns, spread across institutional settings in broader social contexts, are choice objects for ethnographic study. You can visit and live within an algorithmic system, collaborating with the people who work there. For critics and advocates alike, if we want to know algorithms, we may need to live with them.

Works Cited

Amatriain, Xavier and Justin Basilico
2012 Netflix Recommendations: Beyond the 5 stars (Part 1). Netflix Tech Blog. http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html.

Ananny, Mike
2011 The Curious Connection Between Apps for Gay Men and Sex Offenders. The Atlantic. http://www.theatlantic.com/technology/archive/2011/04/the-curious-connection-between-apps-for-gay-men-and-sex-offenders/237340/.

Benedict, Ruth
1946 The Chrysanthemum and the Sword: Patterns of Japanese Culture. Boston: Houghton Mifflin.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest and Clifford Stein
2009 *Introduction to Algorithms* (3rd ed.). MIT Press and McGraw-Hill

Diakopoulos, Nicholas
2014 Algorithmic Accountability Reporting: On The Investigation Of Black Boxes. Tow Center for Digital Journalism. Columbia University. http://towcenter.org/algorithmic-accountability-2/.

Granka, Laura
2010 The Politics of Search: A Decade Retrospective. The Information Society 26:364-374.

Hetherington, Kregg
2011 Guerrilla Auditors: The Politics of Transparency in Neoliberal Paraguay. Durham: Duke University Press.

Hull, Matthew
2012 Government of Paper: The Materiality of Bureaucracy in Urban Pakistan. Berkeley: UC Press.

Karpf, David
2012 Social Science Research Methods In Internet Time. Information, Communication & Society 15(5):639-661.

Kohavi, Ron
2012 Online Controlled Experiments: Introduction, Learnings, And Humbling Statistics. Keynote, Recsys 2012 Dublin.

McGee, Matt
2013 EdgeRank Is Dead: Facebook's News Feed Algorithm Now Has Close To 100K Weight Factors. Marketing Land. http://marketingland.com/edgerank-is-dead-facebooks-news-feed-algorithm-now-has-close-to-100k-weight-factors-55908

Pariser, Eli
2011a The Filter Bubble: What the Internet is Hiding from You. New York: Penguin.
2011b Beware online "filter bubbles." TED. http://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles.html

Strathern, Marilyn
2000 Audit Cultures: Anthropological Studies in Accountability, Ethics and the Academy. Routledge.

Tylor, E.B.
1871 Primitive Culture. John Murray.

Ziewitz, Malte
2013 What does transparency conceal? Privacy Research Group, New York University. http://ziewitz.org/files/Notes%20on%20transparency.pdf