2

The Multiplicity of Algorithms

Consider the following descriptions of what an algorithm is: “a step-by-step instruction of how to solve a task,” “a recipe,” “a form of programmed logic,” “made of people,” “code that tries to accommodate personal interests,” “an application that determines what you see,” “a type of technology,” “an automated filtering mechanism,” “what platforms use to guarantee the best user experience possible,” “an appetizer to make you crave for more of the same,” “magical.” You may already recognize some of these descriptions from standard computer science textbooks, common language, media reports, or from the way in which platforms market their algorithms. Indeed, these descriptions are the accounts of computer scientists, marketers, lay people, critics, journalists, or media users. These are the textbook definitions, tropes, PR slogans, tweets, folk theories, and popular descriptions of algorithms. Yet you may wonder, are these people describing the same thing? What does magic have to do with a recipe? Can something both determine what you see and accommodate your own interests?

This chapter suggests that algorithms are multiple in the sense that there is more than one kind of thing that we call “algorithm.” Not only is this true in the technical sense of there being many different types of algorithms but also in the social sense. When people speak of algorithms they often have different things in mind, different concerns, worries, conceptions, and imaginations. In a more theoretical and philosophical sense, one can talk of such multiplicity as “manyfoldedness, but not pluralism” (Mol, 2002: 84). Whereas plurality assumes that there is one stable thing and different perspectives one can take on that thing, to speak of manyfoldedness implies the idea that algorithms exist on multiple levels as “things done” in practice. In other words, algorithms can be magical *and* concrete, good *and* bad, technical *and* social. Yet, the notion of multiplicity also reminds us that putting it in such binary terms risks perpetuating an overly simplistic understanding of what algorithms are and what they can be. What is at stake, then, is the meaning of a variable existence for the ways in which we understand algorithmic power and politics. While I want to stress a view on algorithms that sees it not as one object (or subject, for that matter) but many, I shall first need to say a few things about the ways in which algorithms are commonly understood.

This chapter offers the conceptual groundwork I believe is necessary to appreciate the multiple levels at which algorithms exist and operate, both technically and theoretically. The first part of this chapter explains what algorithms are in a more technical sense, particularly with regards to machine learning, as most of the algorithms operating in the media landscape depend on systems that are able to learn from data and make predictions. This should provide readers with enough background knowledge to appreciate the question of how algorithms intersect with power and politics on a more philosophical and sociological scale—the concern of the second part of this chapter. While I do not mean to suggest that algorithms can be understood from either a technical or a sociological perspective—as if there exists one single object called an "algorithm" that we might understand from different perspectives—the aim is to show how dynamic both the notion of an algorithm and the concerns they foster are in order to introduce readers to the complexities of algorithmic power and politics. The third part of this chapter uses Foucault's conceptualizations of power to map how power and politics intersect with algorithms as an emerging social concern. I make the argument that algorithms do not merely *have* power and politics; they are fundamentally productive of new ways of ordering the world. Importantly, algorithms do not work on their own but need to be understood as part of a much wider network of relations and practices. The view to which I am subscribing in this book (and which I will further elaborate on in the next chapter), sees algorithms as entangled, multiple, and eventful and, therefore, things that cannot be understood as being powerful in one way only.

A Beginning

HISTORIES AND TECHNICALITIES

Let's take a step back. What exactly do (most) people mean when they invoke the term "algorithm"? According to standard computer science definitions, an algorithm is a set of instructions for solving a problem or completing a task following a carefully planned sequential order (Knuth, 1998). Although the algorithm is a key concept in computer science, its history dates back to the medieval notion of "algorism," understood as a way of performing calculations with natural numbers. Algorism goes as far back as the 9th century when the Persian astronomer and mathematician Abdullah Muhammad bin Musa al-Khwarizmi (circa 780–850) was indirectly responsible for coining the term. When his scripts were translated into Latin in the 12th century, his name was rendered as "Algorithmi."[1] These scripts described the basic methods of arithmetic in the Hindu-Arabic numeral system, which much later formed the basic operations of computer processors (Miyazaki, 2012). As Wolfgang Thomas describes, many paragraphs of the scripts translated from al-Khwarizmi's text "Computing with the Indian Numbers" started with the phrase "Dixit Algorizmi," where algorithm referred to a "process of symbol manipulation"

(2015: 31). During the 17th century, the German philosopher Gottfried Wilhelm Leibniz (1646–1716) added a new dimension to symbolic computation by developing "the vision of calculating truths (true statements)" (Miyazaki, 2012).[2] As far as a the prehistory of algorithms goes, Leibniz provided the groundwork for what later became known as Boolean algebra by "arithmetizing logic" and using if/then conditionals for calculating truth (Thomas, 2015). In Thomas' account, the origin can be traced to Boole (1847) and his theory of Boolean algebra and continues with Frege's 1879 introduction of a formal language in which mathematical statements could be expressed (see Thomas, 2015). For many historians, however, the history of algorithms proper starts with David Hilbert and what is called the *Entscheidungsproblem*, the challenge of whether an algorithm exists for deciding the universal validity or satisfiability of a given logical formula (Sommaruga & Strahm, 2015: xi). In 1936, Alan Turing famously solved this problem in the negative by reducing the problem to a notion of symbolic computation, which later became known as the Turing machine. Although Turing himself scarcely made direct reference to the term "algorithm," contemporary mathematicians such as Alonzo Church (1903–1995) or Stephen C. Kleene (1909–1994) did (Miyazaki, 2012). In tracing the historical origins of the term "algorithm" and its role in the history of computing, Shintaro Miyazaki (2012) suggests that "algorithm" first entered common usage in the 1960s with the rise of scientific computation and higher-level programming languages such as Algol 58 and its derivatives. By the mid 20th century, then, "an algorithm was understood to be a set of defined steps that if followed in the correct order will computationally process input (instructions and/or data) to produce a desired outcome" (Kitchin, 2017: 16).

Perhaps the most common way to define an algorithm is to describe it as a recipe, understood as a step-by-step guide that prescribes how to obtain a certain goal, given specific parameters. Understood as a procedure or method for processing data, the algorithm as recipe would be analogous to the operational logic for making a cake out of flour, water, sugar, and eggs. Without the specific instructions for *how* to mix the eggs and flour or *when* to add the sugar or water, for instance, these ingredients would remain just that. For someone who has never baked a cake, step-by-step instructions would be critical if they wanted to bake one. For any computational process to be operational, the algorithm must be rigorously defined, that is, specified in such a way that it applies in all possible circumstances. A program will execute a certain section of code only if certain conditions are met. Otherwise, it takes an alternative route, which implies that particular future circumstances are already anticipated by the conditional construct of the "if…then statement" upon which most algorithms depend. The "if…then statement" is the most basic of all control flow statements, tasked with telling a program to execute a particular section of code only if the condition is deemed "true." However, in order to be able to test and compute a "false" condition, the "if…then" statements needs to include an "else" statement, which essentially provides a secondary path of executing. In other words, while the "if...then" statement can only compute "true" statements, the "if...then...else" construct will be able to execute an alternate pathway as well.[3] An algorithm essentially indicates *what* should happen *when*, a principle that programmers call "flow of control," which is implemented in source code or pseudocode. Programmers usually control the flow by specifying certain procedures and parameters through a programming language. In principle, the algorithm is "independent of programming languages and independent of the machines that execute the programs" (Goffey, 2008: 15). The same type of instructions can be written in the languages C, C#, or Python and still be the same algorithm. This makes the concept of the "algorithm" particularly powerful, given that what an algorithm signifies is an inherent assumption in all software design about order, sequence, and sorting. The actual steps are what is important, not the wording per se.

Designing an algorithm to perform a certain task implies a simplification of the problem at hand. From an engineering perspective, the specific operation of an algorithm depends largely on technical considerations, including efficiency, processing time, and reduction of memory load—but also on the elegance of the code written (Fuller, 2008; Knuth, 1984).[4] The operation of algorithms depends on a variety of other elements—most fundamentally, on *data structures*.[5] Tellingly, Niklaus Wirth's (1985) pioneering work on "structured programming" is entitled *Algorithms + Data Structures = Programs.* To be actually operational, algorithms work in tandem not only with data structures but also with a whole assemblage of elements, including data types, databases, compilers, hardware, CPU, and so forth.[6] Explaining why he starts his book with data structures and not algorithms, Wirth writes: "One has an intuitive feeling that data precede algorithms: You must have some objects before you can perform operations on them" (2004: 7). Furthermore, Wirth's book title suggests that algorithms and programs are not the same. A program, or software, is more than its algorithms. Algorithms alone do not make software computable. In order to compute, the source code must be transformed into an executable file, which is not a one-step process. The transformation usually follows multiple steps, which include translation as well as the involvement of other software programs, such as compilers and linkers. The "object file" created by the compiler is an intermediate form and not itself directly executable. In order for a program to be executed, another device called a linker must combine several object files into a functional program of executable (or ".exe") files.[7] Software is quite literally the gathering or assembling of different code files into a single "executable." While software might appear to be a single entity, it is fundamentally layered and dependent on a myriad of different relations and devices in order to function. To compute information effectively, then, algorithms are based on particular representations and structures of data. Despite the mutual interdependence of algorithms and data structures, "we can treat the two as analytically distinct" (Gillespie, 2014: 169). Given that my primary interest lies in the *operations* performed on data—what

Wirth defines as algorithms– and the social and cultural implication those operations have, the focus in this book will almost exclusively be on algorithms.

While no agreed-upon definition of algorithm exists, a few aspects have been described as important characteristics. For Donald Knuth (1998), who has written one of the most important multivolume works on computer programming, algorithms have five broadly defined properties: finiteness, definiteness, input, output, and effectiveness. What is generally asked of an algorithm is that it produce a correct output and use resources efficiently (Cormen, 2013). From a technical standpoint, creating an algorithm is about breaking the problem down as efficiently as possible, which implies a careful planning of the steps to be taken and their sequence.

Take the problem of sorting, which is one of the most common tasks algorithms are deployed to solve. A given sorting problem may have many solutions; the algorithm that eventually gets applied is but one possible solution. In other words, an algorithm is a manifestation of a proposed solution. Just as there are multiple ways of sorting a bookshelf in some well-defined order—for example, according to alphabetical order by the author's surname, by genre, or even by the color of the book jacket, different sorting algorithms (e.g., selection sort, merge sort, or quicksort) can be applied for the same task. Anyone who has ever tried to arrange a bookshelf according to the color of the book jacket will probably be able to understand how this specific organizational logic might have an aesthetically pleasing effect but also come with the added practical challenge of finding a particular book by a certain author (unless you have an excellent color memory). This is to say that algorithms, understood as forms of organizational logic, come with specific affordances that both enable and constrain. To use a well-known insight from science and technology studies (STS), such orderings are never neutral. Algorithms come with certain assumptions and values about the world on which they are acting. Compared to the archetypical example used in STS about the inherent politics of engineering and urban planning in the example of Robert Moses' low-hanging bridges described by Langdon Winner (1986), the politics and values of algorithms are a really blatant example of that same logic as they explicitly decide, order, and filter the world in specific ways. Why some people still think of algorithms as neutral, given they are a relatively easy example of an STS value-laden technology, is a different question altogether. Ultimately, algorithms and the models used have consequences. In the example of the bookshelf, a color coding scheme might, for instance, imply that I might become less inclined to read books by the same author simply because I did not facilitate my thinking or decision-making process in terms of genre or author in the first place.

THE LEARNING TYPE OF ALGORITHM

An important distinction needs to be made between algorithms that are preprogrammed and behave more or less deterministically and algorithms that have the ability to "learn" or improve in performance over time. Given a particular input, a deterministic algorithm will always produce the same output by passing through the same sequence of steps. The learning type, however, will learn to predict outputs based on previous examples of relationships between input data and outputs. Unlike a deterministic algorithm that correctly sorts an alphabetized list, many of the algorithms that run the Internet today do not necessarily have one easily definable, correct result. The kinds of algorithms and techniques to which I am referring here are called *machine learning*, which is essentially the notion that we can now program a computer to learn by itself (Domingos, 2015). In contrast to the strict logical rules of traditional programming, machine learning is about writing programs that learn to solve the problem from examples. Whereas a programmer previously had to write all the "if...then" statements in anticipation of an outcome herself, machine learning algorithms let the computer learn the rules from a large number of training examples without being explicitly programmed to do so. In order to help reach a target goal, algorithms are "trained" on a corpus of data from which they may "learn" to make certain kinds of decisions without human oversight. Machines do not learn in the same sense that human do, although many of the algorithms used are based on artificial neural networks are inspired by the structure and functional aspects of the brain's deep architecture (Hecht-Nielsen, 1988). The kind of learning that machines do should be understood in a more *functional* sense: "They are capable of changing their behavior to enhance their performance on some task through experience" (Surden, 2014: 89).

Machine learning algorithms come in many different flavors. Similar to humans, the machine itself learns in different ways.[8] One of the most common ways in which algorithms learn is called *supervised learning*. Essentially an inductive approach to learning, algorithms are given a training set comprising the characteristics that engineers want the algorithm to detect and compare with new data (Flach, 2012). Importantly, the training set includes data about the desired output. When the training data do *not* include data about desired outputs, the approach is called *unsupervised learning*. Often, machine learning algorithms may fall somewhere in between: The data only contain a few desired outputs, which is also called *semi-supervised learning* (Domingos, 2015).[9] Before an algorithm can be applied to learn from data, *models* have to be constructed that formalize the task and goals, so that it can be processed by a computer. For instance, before an algorithm can perform the task of finding the most important news feed stories, models have to be created to represent the relationship between news and relevance. As Mackenzie puts it:

> The techniques of machine learning nearly all pivot around ways of transforming, constructing or imposing some kind of shape on the data and using that shape to discover, decide, classify, rank, cluster, recommend, label or predict what is happening or what will happen. (2015: 432)

In data-intensive environments such as social media, machine learning algorithms have become a standard way of learning to recognize patterns in the data, to discover knowledge, and to predict the likelihood of user actions and tastes. Put another way, machine learning is largely enabled by proliferating data from which models may learn. In the age of so-called big data, having the biggest pool of data available from which to detect patterns is often seen as a competitive necessity. The bigger the database, so the story goes, the better the conditions for algorithms to detect relevant patterns.

Models are central to the understanding of machine learning as "they are what is being learned from the data, in order to solve a given task" (Flach, 2012: 20). The way it works is that a pool of data is first mined (automatically processed) to identify regularities upon which subsequent decision-making can rely. The accumulated set of discovered relationships, then, produce the model, which subsequently "can be employed to automate the process of classifying entities or activities of interest, estimating the value of unobserved variables, or predicting future outcomes" (Barocas & Selbst, 2016: 7). As Joaquin Quiñonero Candela, Director of Applied Machine Learning at Facebook, says about the company's machine learning philosophy:

> 1. Get as much data as you can and make sure it is of highest quality 2. Distill your data into signals that will be maximally predictive—a process called feature-engineering 3. Once you have the most awesome data and tools for feature engineering, keep raising the capacity of your algorithms. (Candela, 2016)

Machine learning, then, is about using data to make models that have certain features. Feature engineering, or the process of extracting and selecting the most important features from the data, is arguably one of the most important aspects of machine learning. While feature extraction is usually performed manually, recent advances in deep learning now embed automatic feature engineering into the modeling process itself (Farias et al, 2016). If the algorithm operates on badly drawn features, the results will be poor, no matter how excellent the algorithm is.

How would feature engineering work in calculating such an abstract notion as relevance? One option might be to consider the frequency and types of content on which a specific user has clicked. However, we might also imagine a scenario in which "most important" is not about the frequency of clicks but, rather, the time spent watching or reading certain content. The point here is that the problem of determining what is "most important" depends on the data and the outcome you want to optimize. As Gillespie suggests, "All is in the service of the model's understanding of the data and what it represents, and in service of the model's goal and how it has been formalized" (2016a: 20). The understanding of data and what it represents, then, is not merely a matter of a machine that learns but also of humans who specify the states and outcomes in which they are interested in the first place.[10] In the case of supervised learning, humans (the data miners and machine learning specialists) first need to specify the target variables (desired states and outcomes), which are then used to define "class labels"—"the different classes between which a model should be able to distinguish" (Barocas & Selbst, 2016: 8). Take the task of filtering spam, an area in which machine learning is commonly applied: The goal of the algorithm is to build an internal computer model that will ultimately allow the machine to make automated, accurate classification decisions (Surden, 2014: 91).[11] However, learning tasks are far from alike. Classifying spam is one of the "easier" tasks as it typically merely operates on binary categories. Either something is classified as spam or it is not. But most of the tasks with which we are concerned in this book have to do with ranking and recommendations that cannot be broken down into binary categories. For example, the question of what constitutes newsworthiness or relevancy is not a straightforward task. As we shall see—particularly in chapters 4 and 6—there is no way to measure newsworthiness directly because the notion is a function of the particular way in which the respective industries and platforms have constructed their systems.[12] Whether it is about ranking content in a feed or classifying email as spam, a useful model depends on the ongoing input of new data to optimize its performance.

The bottom line is that there are many different algorithms that can be used to "impose a shape on the data," as Mackenzie put it. For predictive modelling, for example, well-known algorithmic techniques include: logistic regression models, the Naive Bayes classifier, k-nearest neighbors, support vector machines, random forests and neural networks (Mackenzie, 2015). However, as Bernhard Rieder (2017) points out, something like the Naive Bayes classifier is not yet an algorithm in a more restrictive understanding of the term. Rather, it outlines a common method—an "algorithmic technique"—used to solve specific computational problems. If we want to make it into an algorithm that can be run, we have to further specify it. For random forests, for example, you would have to specify the number of trees, and the number of splits to try in each tree. For neural networks you would have to specify the depth of the hidden layers. So in some sense, we are far away from an algorithm. Yet there are also algorithmic techniques for specifying these so-called hyperparameters.[13] Many pieces of software also come with default parameters. Once you specify these parameters, and you have the data, then it is an "algorithm" in a narrow computational sense, whereas without them it is incomplete. What determines whether to use one technique over another "depends upon the domain (i.e., loan default prediction vs. image recognition), its demonstrated accuracy in classification, and available computational resources, among other concerns" (Burrell, 2016: 5). Take, for example, the problem of face recognition. With more than 350 million photos uploaded onto Facebook every day, face recognition is an important area in which Facebook is using machine learning.[14] Algorithms are trained on a variety of existing data sets to learn how to detect faces in the wild. Depending on the specific task of recognition, however, different algorithms can be deployed. One of the simplest and fastest learning algorithms for this purpose is the nearest neighbor (Domingos, 2015: 179). In order to determine whether an image contains a face, the nearest neighbor algorithm works by finding the image most similar to it in Facebook's entire database of labeled photos. "If" it contains a face, "then" the other one is presumed to contain a face as well. This approach is used, for example, in Facebook's auto-tagging functionality. Given Facebook's vast data pool, the company's engineers are able to train algorithms to detect faces in ways that most photo-sharing sites cannot. Facebook's artificial intelligence team has recently developed a system called *DeepFace*, which is supposedly able to identify faces at a 97.25% accuracy level, which is just slightly worse than the average human score of 97.53% (Taigman et al., 2014). This system uses what is called "deep learning," a technique that currently constitutes one of the leading and most advanced machine learning frameworks available. Deep learning is based on neural networks, a model most widely used in image and speech recognition. Modeled to emulate the way in which the human brain works, neural networks "use different layers of mathematical processing to make ever more sense of the information they are fed" (Condliffe, 2015).[15] In terms of image recognition, a system powered by neural networks, for example, would analyze pixel brightness on one layer, shapes and edges through another layer, actual content and image features on a third layer, and so on. Using different algorithms on every layer to process the information, the system would gain a more fine-grained understanding of the image.[16]

As recent cases of image recognition failures have shown, machine learning is not without its problems, and there is much to be learned from cases in which machine learning goes awry. For example, in May 2015, Google was accused of having a "racist" algorithm when its newly launched Photos app tagged two black people in a photograph as "gorillas" (Barr, 2015). A similar incident happened when the photo-sharing service Flickr, powered by Yahoo's neural network, labeled a person as an "ape" due to the color of his skin. While incidents like these are perfect examples of how the results of machine learning can be very problematic, they are also good examples of how the media, which often neglect to explain why certain inferences were made, frequently reports on algorithms, thus catering to an often shallow understanding of how these systems actually work. My hope is to show how we might better understand the power and politics of algorithms if we try to take a more holistic approach by acknowledging the complexity and multiplicity of algorithms, machine learning and big data. Incidents such as these failed image-recognition tasks point to a core concern about dealing with algorithms, namely, the question of agency, responsibility, and accountability to which I shall return in the next chapter. Suffice it to mention at this point the importance of the training data as another element for understanding the possibilities and limits of machine learning. As Barocas and Selbst point out, "what a model learns depends on the examples to which it has been exposed" (2016: 10). This raises an important question with regards to accusations of a "racist" algorithm: What was the initial data set that Google used to train its image recognition algorithms, and whose faces did it depict? What generalizable signals/patterns was it picking up on?

Understood as sets of instructions that direct the computer to perform a specific task, algorithms are essentially used to control the flow of actions and future events. Sometimes, the flow of events is more or less known in advance, as in the case of algorithms sorting an alphabetized list. More often than not in the world of machine learning, however, the outcome of events remains uncertain. Machine learning algorithms reduce this uncertainty by making predictions about the *likelihoods* of outcomes. Put differently, machine learning is about strengthening the probability of some event happening, based on evolving information. This is also the principle of Bayes' theorem: the "simple rule for updating your degree of belief in a hypothesis when you receive new evidence" (Domingos, 2015: 144).[17] Following Wendy Chun, then, we might understand an algorithm as a "strategy, or a plan of action—based on interactions with unfolding events" (2011: 126). This implies that algorithms do not simply change with the event but are always *in becoming* since events are not static but unfolding. In the case of algorithmically driven sites such as Facebook, users are crucial to the development and maintenance of the underlying coding systems as they constantly feed the system with new data. As Mukund Narasimhan, a software engineer at Facebook, tellingly suggests: "Everything in Facebook is a work in progress." The models Facebook uses to design the system are evolving because the data is changing. This means that the exact ways in which the algorithms work are also constantly tweaked by employees because of the fact that everything else changes (Narasimhan, 2011).

Algorithms do not simply change *with* the event; they also have the ability to change the event. In the era of big data and data mining, algorithms have the ability performatively to change the way events unfold or, at the very least, change their interpretation. A good example of this is the failed Google Flu Tracker, which was operative from September 2011 to August 2013. Often heralded as the prime example of what big data could do, Google's Flu Tracker was designed to predict outbreaks of flu before they happened, based on mining data from their vast troves of search queries. According to Google, they "found a close relationship between how many people search for flu-related topics and how many people actually have flu symptoms" (Walsh, 2014). However, in February 2013, the Flu tracker made headlines because it turned out that it had predicted "more than double the proportion of doctor visits for influenza-like illness than the Centers for Disease Control and Prevention," which had been used as a *de facto* predictor until then (Lazer et al., 2014: 1203). The fact that the tracker was dependent upon Google's search algorithms played a significant part in skewing the results for flu trends. By recommending search terms to its users through its autocomplete feature, Google itself was producing the conditions it was trying to merely describe and predict. As Walsh (2014) put it: "If the data isn't reflecting the world, how can it predict what will happen?"

Algorithms as a Social Concern

Incidents like the Google Flu tracker or the more recent examples of data and algorithmic discrimination outlined above have arguably encouraged many more social scientists and humanities scholars to explore the ethical, political, and social implications that these digital infrastructures and systems have. Since the first cycle of hype and optimism concerning the rise of big data seems to be fading gradually, we are now seeing a growing scholarly interest in what Tarleton Gillespie and Nick Seaver have termed "critical algorithm studies." Over the past decade or so, work spanning sociology, anthropology, science and technology studies, geography, communication, media studies, and legal studies has started to focus critical attention not just on software but algorithms more specifically (Amoore, 2009; Ananny, 2016; Beer, 2009; Cheney-Lippold, 2011, Diakopoulos, 2015; Gillespie, 2014; Introna, 2016; Karppi & Crawford, 2016; Lenglet, 2011; Mackenzie, 2015; McKelvey, 2014; Seaver, 2013; Striphas, 2015; Wilf, 2013; Ziewitz, 2016). Whereas computer scientists typically focus on designing efficient algorithms, a sociological or cultural approach to algorithms is starting to emerge, focusing on what algorithms are actually doing as part of situated practices. Indeed, an algorithmic logic of information processing and dissemination has become pervasive in such fields as finance (Mackenzie, 2015; Pasquale, 2015), transportation (Kitchin & Dodge, 2011), the travel sector (Orlikowski & Scott, 2015), higher education (Introna, 2011; Williamson, 2015), journalism (Anderson, 2013; Diakopoulos, 2015; Dörr, 2016), security (Amoore, 2013; Cheney-Lippold, 2016), surveillance (Braverman, 2014; Introna & Wood, 2004), popular culture (Beer, 2013), and the media industry more generally (Hallinan & Striphas, 2016; Napoli, 2014).[18] Social scientists and humanities scholars are not primarily concerned with the technical details of algorithms or their underlying systems but, rather, with the meanings and implications that algorithmic systems may have. As Gillespie (2016a) suggests, critical algorithm scholars are often more concerned with the algorithm as an *adjective*, understood as the social phenomena that are driven by and committed to algorithmic systems.

While the growing concern over algorithms may be a relatively recent phenomenon in the context of the social sciences and humanities—in part, spurred by the rise of big data—the underlying social concerns have a much longer history. The more pervasive use of algorithms in a wide range of societal sectors and institutions should be seen along a much longer continuum of scholarship concerned with the intersection of technology and society, the social history of computing, and related fields such as new media studies, software studies, platform studies, STS, and human-computer interaction (HCI). Work in these domains has contributed knowledge about the power and politics of computing technologies, software systems, and information infrastructures relevant to our present concern with the role of algorithms in the contemporary media landscape. Take, for example, the growing interest in software as a cultural object of study investigated in the field of software studies (Fuller, 2008). During the past decade or so, social scientists and humanities scholars have called for an expanded understanding of code that extends significantly beyond its technical definitions (Berry, 2011; Fuller, 2008; Kitchin & Dodge, 2011; Mackenzie, 2006). Software studies can be understood as a cultural studies approach to the "stuff of software" (Fuller, 2008), where what counts as "stuff" remains relatively open to interpretation. This conceptual openness is, perhaps, what distinguishes software studies from computer science approaches since what counts as software is seen as a "shifting nexus of relations, forms and practices" (Mackenzie, 2006: 19). This is not to say that scholars interested in software as a cultural object of study dismiss the importance of materiality or technical operations, quite the contrary. While calling for an expanded understanding of the meaning and significance of software as part of everyday life, scholars within software studies also maintain a high level of sensibility toward the technical and functional dimensions of software. In his book *Protocol*, Alexander Galloway makes the case that "it is not only worthwhile, but also necessary to have a technical as well as theoretical understanding of any given technology" (2004: xiii). In order to understand power relationships at play in the "control society," Galloway argues that it is crucial to start with the questions of how technology (in this case, protocols such as HTTP) works and who it works for. In fact, *not* addressing the technical details of software or algorithms as part of a sociological or critical inquiry is seen as problematic (Rieder, 2017: 101). While media scholars differ in their assessment of how necessary coding skills are to a critical understanding of software and algorithms, some technical knowledge is clearly desirable.[19]

If an expanded understanding of software and algorithms does not necessarily mean discarding technical details from analysis, what does it mean? One view, inspired by actor network theory and similar perspectives, takes software and algorithms to be complex technical systems that cannot merely be described as technological alone because their ontological status remains unclear. As Mackenzie argues, software has a "variable ontology," suggesting "that the essential nature of the entity is unstable" (2006: 96). The variable ontology of software means that "questions of when and where it is social or technical, material or semiotic cannot be conclusively answered" (2006: 96). Similarly, Gillespie suggests that "'Algorithm' may, in fact, serve as an abbreviation for the sociotechnical assemblage that includes algorithm, model, target goal, data, training data, application, hardware—and connect it all to a broader social endeavour" (2016a: 22). Moreover, as Seaver points out, "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" (2013: 10). Another way of answering the question of what algorithms are beyond their technical definition is to see them as inscriptions of certain ideologies or particular ways of world-making (Goodman, 1985). For example, scholars have analyzed the algorithmic logic underpinning search engines and high-frequency trading in terms of how they can be said to advance a capitalist ideology (Mager, 2012; Snider, 2014). Another option, not unlike the networked view, is to emphasize the ways in which algorithms are "entangled in practice" (Introna, 2016; Gillespie, 2014; Orlikowski & Scott, 2015). This is the notion that algorithms do things as part of what else they are entangled with. As Gillespie writes, "algorithms are built to be embedded into practice in the lived world that produces the information they process" (2014: 183). Researchers have drawn on notions of sociomateriality, performativity, or ideas of entanglement to focus their attention not on what algorithms do in technical terms but on what they do in terms of constituting and being constituted in and through practice. What matters is not so much the precise instructions or the theorem underlying the algorithm, but how it is incorporated within specific sociomaterial practices (Introna, 2016).

The notion of algorithm as adjective suggests that, when social scientists and humanities scholars invoke the term "algorithmic," what they are really concerned with "is not the algorithm per se but the insertion of procedure into human knowledge and social experience" (Gillespie, 2016a: 25). The insertion of procedurality and quantification into human experience has been noted in various recent accounts on culture, aesthetics, and knowledge production. Cultural theorist Ted Striphas (2015), for example, writes about the emergence of an "algorithmic culture," which he takes to be the ways in which platforms such as Netflix and Amazon are seen to alter the way in which culture has traditionally been practiced, experienced, and understood.[20] Furthermore, Mark Lenglet (2011) describes how the financial world has become algorithmic not only by having algorithmic procedures inserted into trading but also through the ways in which algorithms now occupy the minds of traders, making them act and react in certain ways. As I will discuss in chapter 5 and 6, this is not unique to the financial world but is evident in everything from social media users orientation toward platforms through to the institutional contexts of journalism where algorithms are inserted into both journalistic practices and discourse. Even identity seems to have become algorithmic, as evidenced by the "shift from offline to online marketing" (Cheney-Lippold, 2011: 175). As seen from the perspective of many sociologists, cultural theorists, and media scholars, what is interesting about algorithms is not necessarily the same as what interests a computer scientist or an engineer. While an algorithm is still a set of instructions used to solve a computational problem, the valences of the term obviously differ in mathematics and sociology. However, as Gillespie points out, "to highlight the mathematical quality is not to contrast algorithms to human judgement. Instead it is to recognize them as part of mechanisms that introduce and privilege quantification, proceduralization, and automation in human endeavors" (2016a: 27).

The social concerns voiced about algorithms today are not necessarily new. As mechanisms of quantification, classification, measurement, and prediction, algorithms are as much imbued in the history of computation and software engineering as they are in the history of statistics, accounting, and bureaucratization. As such, the historical and cultural contexts of algorithms intersect with the social history of calculation and ordering of various types, including the history and politics of statistical reasoning and large numbers (Desrosieres & Naish, 2002; Foucault, 2007; Hacking, 2006; Power, 2004); practices of quantification, numbering and valuation (Callon & Law, 2005; Espeland & Stevens, 2008; Verran, 2001); the cultural logic of rankings and ratings (Espeland & Sauder, 2007; Sauder & Espeland, 2009); and ideas of critical accounting and auditing (Power, 1999; Strathern, 2000b). Moreover, contemporary concerns about the power of algorithms can also be seen as an "extension of worries about Taylorism and the automation of industrial labor" (Gillespie, 2016a: 27) or "the century long exercise by media industries to identify (and often quantify) what's popular" (Gillespie, 2016b). Clearly, algorithms need to be understood as part of a historical lineage. Indeed, we might say that the "avalanche of numbers" (Hacking, 1991, 2015), which occurred as nation-states started to classify and count their populations in the 19th century, forms a general backdrop for an understanding of algorithms in the era of big data. This specific historical lineage, however, if we were to carry it all the way through, would also require us to travel a complex and disjunctive route via the social history of census, punch cards, bureaucratization and the rise of industrial society and factory work, wartime machinery and the rise of computers, databases and the automated management of populations, before arriving at the contemporary milieu of "making up people" (Hacking, 1999) through data mining and machine learning techniques. This is the genealogy that tells the story of managing populations—most notably theorized by Michel Foucault in his *College de France* lectures (Foucault, 2007, 2008). As Foucault (2007: 138) points out, statistics means etymologically "knowledge of the state." The task for those who govern, then, is to determine what one needs to know in order to govern most effectively and how that knowledge is to be organized. If the collection of data through the numbering practices of statistics and the corporate data mining of recent years makes algorithmic operations possible, the algorithms themselves (understood in the technical sense) give shape to otherwise meaningless data. While the significant power and potential of big data (the quantity of information produced by people, things, and their interactions) cannot be denied, its value derives not from the data themselves but from the ways in which they have been brought together into new forms of meaningfulness by the associational infrastructure of the respective software systems in which algorithms play a key role.

Power of Algorithms

The starting premise for the book is the observation that algorithms have become a key site of power in the contemporary mediascape. During the past decade or so, social scientists and humanities scholars have started to explore and describe the increased presence of algorithms in social life and the new modes of power that these new "generative rules" entail (Lash, 2007; Beer, 2009; Cheney-Lippold, 2011; Gillespie, 2014; Diakopoulos, 2015). As Scott Lash argues, "[a] society of ubiquitous media means a society in which power is increasingly in the algorithm" (2007: 71). To say that algorithms *are* powerful, *have* power, animate, exert, or produce power needs some qualification and explanation. The concept of power is one of the most contested and important terms there is. While it is difficult to provide a short answer to the fundamental question of what power is, Michel Foucault's scholarship provides one of the most comprehensive avenues for a nuanced and multifaceted understanding of the term. For Foucault, power was never just one thing but, fundamentally, about different forms of relations. Throughout his career, Foucault changed and modified his ideas of what power is, coining analytically distinct conceptions of power sensitive to specific contexts, institutions, objects of knowledge, and political thought.[21] Generally speaking, Foucault identified three levels of power relations, which he termed strategic games between liberties, domination, and governmental technologies. As Foucault elaborates in an interview:

> It seems to me that we must distinguish the relationship of power as strategic games between liberties—strategic games that result in the fact that some people try to determine the conduct of others—and the states of domination, which are what we ordinarily call power. And, between the two, between the games of power and the states of domination, you have governmental technologies. (Bernauer & Rasmussen, 1988: 19)

The first level—strategic games between liberties—is a ubiquitous feature of society and human interaction. From this very broad conception, it follows that there is nothing outside power relations because power conditions the very existence of society. Although Foucault is mostly associated with conceptualizing power as immanent to the modern social productive apparatus, he also maintained a notion of power that acknowledged the asymmetrical relationships of the subordinate position of certain individuals and groups. Yet, domination was never seen as default form of power but rather the exception. The third notion of power relations—government—refers to a more systematized and regulated form of power that follows a specific form of rationality. Foucault's notion of government demarcates power from domination by seeing power, first and foremost, as a form of guidance and "shaping the field of possible action of subjects" (Lemke, 2012: 17). In many ways, these three levels of power can be seen in the various ways in which the power of algorithms has recently been framed within the literature. While most authors do not explicitly frame their notion of algorithmic power in Foucauldian terms, connecting the discussion back to Foucault might help analytically to distinguish what is at stake in making a claim about the powerful role that algorithms have in society today.

Like the immanent form of power described by Foucault, Scott Lash sees new forms of capitalist power as "power *through* the algorithm" (Lemke, 2012: 17, emphasis mine). This is a form of power that works from below, not a power-over as it were. As such it becomes indistinguishable from life itself by sifting into "the capillaries of society" (Lash, 2007: 61). In his seminal article on algorithmic power in the age of social media, Beer builds on Lash's notion of "post-hegemonic" forms of power to argue that the algorithms underpinning social media platforms "have the capacity to shape social and cultural formations and impact directly on individual lives" (2009: 994). Drawing on the example of Last.fm, a music-sharing platform, Beer argues that the power of the algorithm can be seen in the ways that the platform provides users with their own taste-specific online radio station. Power, in others words, stems from the algorithm's capacity to "shape auditory and cultural experiences" (2009: 996). This notion of power through the algorithm does not treat power as hierarchical or one-directional but, rather, as immanent to life itself. Algorithmic power in this sense can be understood as a force, energy, or capacity of sorts (Lash, 2007). This way of understanding power as immanent is, perhaps, most famously reflected in Foucault's notion of power as an omnipresent feature of modern society in which power relations are not seen as repressive but as productive (Lemke, 2012: 19).[22]

Others have argued that algorithms have an intrinsic power to regulate social lives through their "autonomous decision-making" capacity (Diakopoulos, 2015: 400). Here, power seems to be located *in* the mechanics of the algorithm. According to Nick Diakopoulos, algorithmic power stems from the "atomic decisions that algorithms make, including *prioritization*, *classification*, *association*, and *filtering*" 2015: 400, emphasis in the original). As such, algorithms exert power by making decisions about the ways in which information is presented, organized, and indicated as being important. As filtering devices, algorithms make decisions about what information to include and exclude, constituting a new form of gatekeeping (Bozdag, 2013; Helberger et al., 2015; Napoli, 2015).[23] In contrast to the rhetoric surrounding the early days of the Web, which often heralded the Internet's potential for doing away with hierarchy by giving everyone a voice, scholars now worry that algorithms are assuming gatekeeping roles that have a significant effect on the way public opinion is formed (Just & Latzer, 2016). Worries about algorithms diminishing the democratic potential of the public sphere—for example, by creating filter bubbles (Pariser, 2011; Zuiderveen et al., 2016) or manipulating what information is shown to the public in the first place (Tufekci, 2015)—are frequently framed in more traditional terms as forms of domination or power seen as hierarchical and top-down. In this sense, algorithms are seen as *having power over* somebody or something. As Frank Pasquale writes in his recent book *The Black Box Society*, search engines and social networks, by way of their capacity to include, exclude, and rank, have "the power to ensure that certain public impressions become permanent, while others remain fleeting" (2015: 14). In Foucault's terms, this would be power seen as a form of political, social and economic domination, where one entity prevents another from seeing or doing something. As Pasquale puts it, "we have given the search sector an almost unimaginable power to determine what we see, where we spend, how we perceive" (2015: 98). However, framing algorithms as having power over someone or something risks losing sight of the human decision-making processes and programming that precedes any algorithmic operation. When Pasquale notes that we have given power to the search sector, he does not simply mean the algorithms running Google. Yet, there is a tendency to use algorithm as a placeholder for a much more distributed form of power. As I argue in this book, critical inquiry into algorithmic forms of power and politics should always extend any claim of algorithms *having* power. As I will elaborate in in the next chapter, the question of whom or what power most obviously belongs to cannot be conclusively answered. Instead, what can be examined are algorithms *in* practice, the places and situations through which algorithms are made present and take on a life of their own.

Claims about algorithmic decision-making, which are often accompanied by subsequent calls for greater accountability and regulation (Diakopoulos, 2015; Pasquale, 2015), are part of a much longer continuum of concerns about the ways in which technology can be said to have politics. According to this view, technology is never neutral but always already embedded with certain biases, values, and assumptions. At least since Winner's (1986) influential arguments about Moses' low-hanging bridges, scholars have followed suit in advocating the necessity to consider the politics of artifacts, particularly by attending to the values in design and the moral import of those design choices.[24] Introna and Wood (2004), for example, argue that facial recognition systems are political in the sense that these algorithmic forms of surveillance carry certain assumptions about designated risks in the very design of the system. As Introna and Wood explain, the politics is mostly implicit, "part of a mundane process of trying to solve practical problems" (2004: 179). Every artifact implicating human beings, including algorithms, always carries certain assumptions and values about how the world works. Take the design of an ATM: "if you are blind, in a wheelchair, have problem remembering, or are unable to enter a PIN, because of disability, then your interest in accessing your account can be excluded by the ATM design" (2004: 179). The point is not that designers of ATM machines consciously or deliberately exclude people in wheelchairs from using an ATM but that the ways in which systems are designed always involve certain exclusionary practices that sometimes appear to be a more or less coherent and intentional strategy despite nobody "authoring" it as such (Introna and Nissenbaum, 2000).

The question of intentionality—whether someone explicitly authors an artifact to function in a particular way or whether its functioning is emergent—gets even more complicated with respect to machine learning algorithms. These algorithms are able to improve performance over time based on feedback. In addition, machine learning algorithms "create an internal computer model of a given phenomenon that can be generalized to apply to new, never-before-seen examples of that phenomenon"

(Surden, 2014:93) without being explicitly programmed to do so. How should we think about the politics of artifacts when the design of the system is continuously performed by the system itself? Returning to the example of the failed image recognition in the case of Google's Photos app, the nature of the training data might be just as important to consider as the algorithm itself. As Gillespie points out:

> The most common problem in algorithm design is that the new data turns out not to match the training data in some consequential way [. . .] Phenomena emerge that the training data simply did not include and could not have anticipated [. . .] something important was overlooked as irrelevant, or was scrubbed from the training data in preparation for the development of the algorithm. (2016a: 21)

Indeed, as Introna and Wood contend, it seems "we cannot with any degree of certainty separate the purely social from the purely technical, cause from effect, designer from user, winners from losers, and so on" (2004: 180). This does not mean, however, that taking a "value in design" approach in the study of machine learning algorithms is not a viable option. To the degree that algorithms are, in fact, capable of labeling black people as "gorillas" or to score black people as predominantly high-risk in committing a future crime (Angwin et al., 2016), they ought to be scrutinized for the kinds of values and assumptions underlying their function. While many scholars and policymakers have worried about the discriminatory capabilities of algorithms, calling for more transparency on part of the companies involved is only part of the solution. What many of the "algorithms gone awry" cases suggest is the importance of seeing the results as reflections of more fundamental societal biases and prejudices. This is to say that, if the machine that is supposed to compute the likelihood of future crimes is fed statistical data tainted by centuries of racial bias materialized in police reports, arrests, urban planning, and the juridico-political systems, it would be misleading only to talk about the power of algorithms in producing such risk assessments. Instead of locating predictive power *in* the algorithm (narrowly defined), we may think of algorithms as what Foucault calls governmental technologies (Lemke, 2001; Miller & Rose, 1990).

Foucault used the notions of government and governmentality to analyze the forms of power and politics involved in the shaping and structuring of people's fields of possibility. While Foucault used the terms government and governmentality somewhat interchangeably, especially in his later writings, "government" broadly refers to the "conduct of conduct," whereas governmentality refers to the modes of thought, or rationalities, underlying the conduct of conduct (Foucault, 1982; 2007; 2010; Lemke, 2001; Rose, 1999).[25] As Lemke remarks, "the problematic of government redirects Foucault's analytics of power" (2012: 17). It is a notion of power that moves beyond relations of consent and force and, instead, sees power relations in the double sense of the term "conduct" as both a form of leading and "as a way of behaving within a more or less open field of possibilities" (Foucault, 1982: 789). Government is about "the right disposition of things" (Foucault, 2007: 134), which concerns "men in their relationships, bonds, and complex involvements with" various things such as resources, environment, habits, ways of acting and thinking (Foucault, 2007: 134).[26] By way of example, Foucault asks what it would mean to govern a ship?

> It involves, of course, being responsible for the sailors, but also taking care of the vessel and the cargo; governing a ship also involves taking winds, reefs, storms, and bad weather into account. What characterizes government of a ship is the practice of establishing relations between the sailors, the vessel, which must be safeguarded, the cargo, which must be brought to port, and their relations with all those eventualities like winds, reefs, storms and so on. (Foucault, 2007: 135)

Extending the analytics of government to an understanding of algorithmic power implies a concern for the ways in which things are arranged and managed to guide people in a certain way. For Foucault, this organizing power has a crucial technological dimension since the conduct of conduct is achieved through various technical means.[27] Introna (2016), for example, explores this dimension of algorithmic power by considering how plagiarism algorithms are used to govern academic writing practices. He makes the argument that algorithms have the power to enact particular governed subjects, exemplified in the way that the plagiarism software enacts "a particular understanding of originality and plagiarism, as well as subjects who conceive of 'good' writing practice as the composition of undetectable texts" (Introna, 2016: 20). As with Foucault's earlier notion of disciplinary power, a concept that will be further discussed in chapter 4, the notion of government and governmentality strongly hinges on subjectivation as a core principle of power. The making of subjects and subject positions in Foucault's work, frames power as something that is productive and generative.

Power as such does not exist; it only exists "when it is put into action" (Foucault, 1982: 788). For an understanding of algorithmic power, this implies that algorithms do not *have* or possess power. Instead, algorithms enact "the possibility of conduct and putting in order the possible outcome" (Foucault, 1982: 789). "Governmentality" does not "force people to do what the governor wants" but, rather, works by assuring "coercion and processes through which the self is constructed or modified by himself" (Foucault, 1993: 204).[28] I want to suggest that we conceive of government and governmentality as particularly helpful concepts in understanding the power of algorithms. Algorithms do not simply have power in the possessive sense; they constitute "technologies of government." Just as statistics enabled the governing of populations in Foucault's account of security, algorithms operate as instruments of government to direct the flow of information and the practices of users "to this or that region or activity" (Foucault, 2007: 141).

To conceive of algorithms as *technologies* of government in the Foucauldian sense, however, is not to restrict analysis of algorithms to the material domain. As Lemke writes, "an analytics of government operates with a concept of technology that includes not only material but also symbolic devices" (2012: 30). Discourses and narratives are "not reduced to pure semiotic propositions; instead they are regarded as performative practices" and part of the mechanisms and techniques through which the conduct of conduct is shaped. Throughout the book, the different ways in which algorithms operate as technologies of government will be both explicitly and implicitly addressed. In chapter 4, Facebook's news feed algorithm is explicitly framed as an architectural form that arranges things (relationships between users and objects) in a "right disposition." Chapter 5 highlights the ways in which narratives and beliefs about algorithms constitute a double articulation, shaping both the conduct of individuals and the algorithm through feedback loops. Chapter 6, meanwhile, considers how algorithms are used to govern journalism in a digital age, looking at how algorithms do not just enact a particularly understanding of journalism but also through the ways in which algorithms now occupy the minds of news professionals, making them act and react in certain ways.

Targeting the Algorithm as the Target

A final word needs to be said about the choice of making the algorithm the target in the first place. While I maintain a broad conception of algorithms throughout the book as a way of talking about the imbrication of computer rules and instructions on everyday life, it is worth noting that using the algorithm as the target is not without its problems. As Ian Bogost has argued, there is a certain danger of theologizing the algorithm by singling it out as the target for describing contemporary culture. While I agree with Bogost (2015) that algorithms are not simple, singular systems but, rather, multipart and complex, involving a "varied array of people, processes, materials, and machines," it does not discredit algorithms as objects of study and phenomena of interest, quite the contrary. Precisely *because* of their entangled and complex nature, there is an obvious need to disentangle their meanings and ways of acting in the world.

What I want to do in this book is to explore and understand better what it means to have algorithms interwoven in the social fabric of the contemporary media landscape. Targeting the algorithm is primarily about developing a better understanding of the rhetorical registers that these mechanisms of proceduralization have. That is, precisely because the "algorithm has taken on a particularly mythical role in our technology-obsessed era" (a "sloppy shorthand" for something much more complex, as Bogost puts it), not-so-sloppy descriptions are called for. When platforms such as Twitter and Instagram publicly announce that they are introducing an "algorithmic timeline" or when the media are repeatedly reporting on incidents of "algorithmic discrimination" and bias, there is also an increasing need to examine not just what the term means but, more importantly, how and when "algorithms" are put to use as particularly useful signifiers and for whose benefit. In the way Katherine Hayles (2005: 17) sees computation as connoting "far more than the digital computer" and "not limited to digital manipulations or binary code," this book sees algorithms as much more than simply step-by-step instructions telling the machine what to do.[29] When a London bookstore labels a bookshelf containing employees' book recommendations their "human algorithm" or when the BBC starts a radio show by announcing that "humans beat robots," there is clearly something about the notion of an algorithm that seems to inform the way we think and talk about contemporary culture and society that is not just strictly confined to computers.[30]

Algorithms, I want to suggest, constitute something of a cultural logic in that they are "much more than coded instruction," drifting into the ways in which people think and talk about everything from the economy to knowledge production to culture. Algorithms exist on many scales, ranging from the operationality of software to society at large. However, in this book, I am less interested in asserting what algorithms are, as if they possess some essence that can be clearly delineated. What is of interest is the "ontological politics" of algorithms, the sense that "conditions of possibility are not given" but shaped in and through situated practices (Mol, 1999: 75). Algorithms are seen as multiple. This is to say that there is no one way in which algorithms exist as a singular object. By positing the multiple nature of algorithms, the intention is to take their manyfoldedness seriously.[31]

Drawing on Mol's work in *The Body Multiple* (2002) in which she argues that there is no essence to what the disease arthrosclerosis is—only ever different *versions* that are enacted in particular settings, I argue that algorithms never materialize in one way only. This does not mean, however, that there cannot be a singular object called an algorithm or that algorithms are nothing but relations. Like the diseases studied by Mol or the aircraft studied by John Law, algorithms oscillate between multiplicity and singularity. As Law puts it: "Yes, atherosclerosis. Yes, alcoholic liver disease. Yes, a water pump [. . .] And yes, an aircraft. All of these are more or less singular but also more or less plural" (2002a: 33). Similarly, we might say: Yes, a neural network. Yes, a code written in C++. Yes, PageRank. Yes, a conversation about how we organize our lives. All of which are more or less singular, more or less plural. In contrast to Mol and Law, however, this book is not about a specific object such as one specific aircraft (Law is specifically concerned with the military aircraft TSR2), disease (i.e., anemia), or algorithm (i.e., PageRank). The singularity and multiplicity in question are the result of analytical cuts. Sometimes, the algorithm might appear as a singular entity—for example, in chapter 4 on the Facebook news feed algorithm.

At other times, the algorithm is less identifiable; yet, the question of what and where it is looms large. What is at stake, then, in addressing the ontological politics of algorithms is not so much an understanding of what exactly the algorithm is or the moments in which it acts (although this is important, too) but, rather, those moments in which they are *enacted* and made to matter as part of specific contexts and situations.