Algorithms, Performativity and Governability (**** **early draft** ****)

Lucas D. Introna, Lancaster University (l.introna@lancaster.ac.uk)

Introduction

[still need to write this]

Algorithms

Algorithms: what are they and what they do?

At its most basic level an algorithm is merely the set of instructions fed into the machine to solve a well-defined problem. Generally we would differentiate between the algorithm (the set of instructions) and its implementation in a particular source language. Algorithms usually express the computational solution in terms of logical conditions (knowledge about the problem) and structures of control (strategies for solving the problem) – leading to the following definition that all computing students are taught: algorithms = logic + control (Kowalski 1979). A classic example of an algorithm is the bubble sort algorithm, designed to sort a list of unsorted numbers.

The algorithm for a bubble sort: Starting from the beginning of the list, compare every adjacent pair, swap their position if they are not in the right order (the latter one is smaller than the former one). After each iteration, one less element (the last one) is needed to be compared until there are no more elements left to be compared.

In the C++ implementation of bubble sort, in the box on the right, we see, for example, expressions of *logic* in the form of 'if' statements and expressions of *control* in terms of 'for' statements. Together they express the instructions to implement the bubble sort algorithm above. A skilled programme can take an algorithm (as the one above) an express it in a computing language such as, for example C++ or Java.

Implementation of bubble sort in C++

```
void bubblesort3( int * a , int n)
{
   int temp,swaps;
   for( int i = 0 ; i < n - 2 ; i++ )
   {
      swaps=0;
      for ( int j = 0 ; j < n - i - 1 ; j++ )
      {
         if ( a[j] > a[j + 1] )
         {
            temp = a[j];
            a[j] = a[j + 1];
            a[j + 1] = temp;
            swaps++;
         }
      }
      if( swaps == 0 )
      {
         break;
      }
   }
}
```

This C++ source code needs to be translated into object code (usually a machine code, also called a machine language) in order to be 'executable' by

a computer. The exact nature of the object code depends on the specific type of CPU (central processing unit) of the computer. Machine code is usually not directly readable by humans.

For a problem to be solved (in computational terms) it needs to be able to be expressed algorithmically, that is, in terms of logic and control. The skilful programmer is able to take a problem and express it algorithmically (often in what is called pseudo code). Once expressed as such it can be translated into source code to be executed by a computer. Thus an algorithm (or subsequent source code) expresses problems in a way that they can be solved by a computational machine. Or, algorithms are solutions to problems expressed in ways that make them amenable to computational implementation. Before we continue we need to note two things. First, not all problems can be expressed algorithmically. This means that computer programmers tend to search for ways to express problems in computational terms (and also tend *not* to favour solutions or approaches that cannot be expressed algorithmically). This is especially true in terms of the source language that they are most skilled at. Ullman (1997a, 110) argues that rapid development of the technological landscape means that programmers often live in a situation of more or less 'ignorance'. "This is not often talked about: we computer experts barely know what we are doing. We're good at fussing and figuring out. We function in a sea of unknowns… Over the years, the horrifying knowledge of ignorant expertise became normal, a kind of background level of anxiety.' In this world of more or less ignorance there is often a sort of implicit but powerful computational reductionism at work in the practice of programming. Namely problems are reduced to what can be expressed algorithmically (and especially, in terms of the technical expertise available to the programmer). This reductionism can have significant performative consequences *for what software actually does*, as I hope to show below.

Second, there is an inevitable chain of translations in which the algorithm becomes implemented and progressively invisible, and diffused—i.e. becomes more or less inscrutable. And, as Latour (2005) never tires to remind us, every translation is also a transformation—see also Harman's (2009) discussion of Latour in this regard. It is a non-trivial question to ask where or what the algorithm/software is—as Wendy Chun (2008) has demonstrated. For example if we want to verify that the software in an electronic voting system does indeed allocate votes correctly. What will we verify? Will we look at the algorithm, the source code or the object code (if we can read it)? Or will we verify its operation? If we can verify the operation, how will we know that the object code running now (when we test it) is the same object code that will be running on the night of the election – we cannot inspect it directly, especially not in operation? Even if we can read the source code, will we be able to spot an error when we have 50 million lines of source code to inspect (apparently Windows Vista consist of 50 million lines of source code). This problem of where the algorithm is (in its actuality) and how to know what it does is made even more complex with the advent of machine learning algorithms based on neural nets. These algorithms adapt themselves through experience (exposure to a specific data set). For

example, many facial recognition systems are based on machine learning algorithms. In such cases it is very difficult to know what the algorithm considers when identifying a face, for example (L. D Introna and Wood 2004). In many ways the algorithm is a black box, which when opened simply introduces more black boxes, which when subsequently opened simply introduces more black boxes, and so forth. The metaphor of an onion that is peeled away, layer by layer is often used. Some might claim that programming approached such as object orientation has merely intensified this invisibility and inscrutability (Fuller 2008, 203). It is therefore not surprising that representatives of Google and other organisations respond that it is difficult for them to say exactly what it is that their algorithms *actually do*. I will take up this matter in much more detail below. However, before we move on it might be important to say why algorithms seem to concern us in the way they do.

The supposed power of algorithms (why do the concern us?)

Algorithms (implemented as software) are said to be powerful, and dangerous, because they operate under the surface, or in the background. We cannot directly inspect them (as object code) or, in many cases, understand them as source code—as was suggested above. Design decisions, encoded and encapsulated in complex nests of logical and control statements—rules within rules within rules—enact (in millions of lines of source code) our supposed choices based on complex relational conditions, which after many iterations of 'bug fixing' and 'tweaking' *even the programmers no longer understand*. As Ullman (1997a, 116–177) observes: "The longer the system has been running, the greater the number of programmers who have worked on it, the less any one person understands it. As years pass and untold numbers of programmers and analysts come and go, the system takes on a life of its own. It runs. That is its claim to existence: it does useful work. However badly, however buggy, however obsolete - it runs. And no one individual completely understands how" (emphasis added). Once encoded, these design decisions (or rather the outcomes of the initial hacking and tweaking) embedded in these multifarious encoding entanglements withdraws into the background and are hardly ever revisited—not even if they break down, patching and workarounds would normally suffice.

Yet some have argued that these encoded geographies (Graham 2005) seem to configure and circumscribe us and our lives in more or less significant ways. Defining what is relevant and what is not, what needs attending to and what not—legitimating particular ways of being whilst simultaneously delegitimizing (or rendering more or less obscure) equally valid alternatives. Or as Lessig (2006, 79) argues: "As the world is now, code writers are increasingly lawmakers…." In and through these encoding practices of programmers and system designers an encoded 'technological unconscious' is emerging which sustains a "presence which we cannot access but which clearly has effects, a technical substrate of unconscious meaning and activity" (Thrift and French 2002, 312).

If algorithms were indeed so powerful then should we not merely focus on them? For example should we not mandate Google to correct their ranking algorithm so that it is fair,

get Tripadvisor to reveal the specific nature of its ranking algorithm, and so forth? The problem with such an approach is that we tend to award a substantial amount of agency to algorithms without appreciating that algorithms function merely as one 'actor' in a total assemblage of actors – what about the user? As Mackensie (2006, 10) suggests: "Deciding where code locates agency in software, how code distributes agency and who or what else possesses agency becomes complicated at certain moments. The critical agent in a situation becomes hard to identify, ambivalent, even threatening…. Does agency center on the programmer who tinkered with lines of code to make them do something she imagined? Does a program act as a vehicle for the agency of someone who runs it, "the user"? Or does it lie with the corporation whose enforcement of intellectual property rights radically curtails different uses of software?" He goes on to suggest that software operates in a mileu in which there are "bodies, things, systems, conventions and signs (and this is virtually everywhere)" and in which "agency distributes itself between people or between people and things in kaleidoscopic permutations. Agency exists in events and ensembles that generate different attributions".(Mackenzie 2006, 10). Thus, the question of who does what, where and in what way is not a trivial one. This is where I want to focus. However, before we proceed I want to say something about the claim that software is a particularly powerful agent because it is 'executable.'

Being executable is a significant claim, and if true, a very significant reason for concern, especially if they are as inscrutable as is being suggested. If software code is directly executable then it means these algorithmic systems can operate 'automatically' (in the background) without the need of human intervention, as it were. Galloway (2004) argues, for example, that software code is very different to ordinary language as "[software] code is the only language that is executable." In a similar manner Ullman (1997b) suggests that "We can use English to invent poetry…. In programming you really can't. ... a computer program has only one meaning: what it does…. Its entire meaning is its function". Hayles (2005, 50) in her essay "Speech, Writing, Code" tends to agree with these claims. She suggests that "code that runs on a machine is performative in a much stronger sense than that attributed to language" since, she argues, "the performative force of language is...tied to the external changes through complex chains of mediation." Whilst one might agree with the general point one could equally argue that these distinctions, between software code and ordinary language, for example, are distinctions of degree rather than distinctions of kind. I would suggest that all code, to be code, must be 'executable'—otherwise it would not translate/transform agency (Latour 1988). Legal code, to translate the agency of the legislative body, also needs to be executable. A recipe for baking a cake, to be a 'recipe' rather than merely some suggestions, needs to be executable. The difference between these varies types of 'executability' is the nature of the necessary constitutive conditions for such execution. Indeed Wittgenstein (2001) would suggest to Ullman that the meaning of all language, not just software, "is its function." To be 'executable' all codes must necessarily conform to their constitutive conditions as code—a move in chess is only 'a

move' if it conforms to the rules of chess, a hammer can only 'hammer' if it is used in the appropriate manner, java will only be a 'programming language' if it conforms to the Java syntax (and much more besides). Chun (2008) makes this same argument when arguing against the supposed 'executable' nature of software code: "source code is never simply the source of any action; rather, source code is only source code after the fact: its effectiveness depends on a whole imagined network of machines and humans" (299). I would suggest that we need to understand the constitutive conditions of algorithms, not only in the sense of what is necessary for them to 'run', but, more importantly, for them to do what they do in the situated practices within which they operate as the algorithms that they are becoming – that is, in their performativity, which is where I want to focus for the remainder of this paper.

Performativity, and the *doing* of algorithms

> That *how* an actual entity becomes constitutes *what* that actual entity is... Its 'being' is constituted by its 'becoming'. This is the principle of process. (Whitehead 1978, 23)

The ontology of becoming

What do algorithms do? Algorithms, for the programmer, solve concrete computing problems by defining a set of formal procedures to be performed in order to accomplish a task in a finite number of steps. Programmers solve computational problems. Software, for users, do not solve computing problems, rather they are components of situated social practice (something in order to do something within a situated practice). For Google programmers the ranking algorithms is a computational problem of finding and using available data to rank search results in such a way that the top ranked result would be most relevant given a set of search terms (or phrases) *and* preventing search engine optimisers (SEO) from manipulating the data to push their desired result up the list of results, *and* optimising the use of scarce computing resources, and much more besides. For users ranking is about finding what they looking for as part of what they are doing, such as researching a topic, trying to buy a product, just browsing, etc. Indeed they are never just 'searching' (in the same why we are never just 'talking'). Rather, they are searching in order to do something specific. What the algorithm 'does', in these two cases, are very different things. The 'doing' of the algorithm, for the different actors, will overlap in some way, of course, but not entirely. One might say that it is the interaction between the non-overlapping doings that is most interesting to try and understand, especially in terms of its performativity. It is in this regard that MacKenzie (2006) suggests that "[a]gency exists in events and ensembles that generate different attributions." Moreover, in each of these cases the activity of the various actors will 'inherit', via the algorithm, some form of 'doing' (knowingly or unknowingly) of the other actors involved. For example, the users will 'inherit' (probably unknowingly) some of the attempts of the Google programmers to outsmart the SEO attempts at manipulation as they go about researching, shopping, browsing, etc. Finally, as these actors draw upon the varieties of doing of the algorithm, as

part of their own situated practices, there will be *performative* outcomes that is not strictly speaking 'in' the algorithm, 'in' the user, or 'in' any of the actors as such, but are produced through the on-going and situated 'doing' by all the actors as part of a socio-material assemblage. I believe it is these performative outcomes that are really the issue to the considered when thinking about the governing of algorithms. Before we consider a detailed example I would like to say something about *performativity*.

Performativity is a particular perspective on the nature of *interaction* which is the logical outcome of a particular ontological position of *becoming* (also known as process philosophy) as opposed to an ontology of *being* (Whitehead 1978; Barad 2007; Butler 1990; Latour 1988). In simple terms an ontology of becoming suggests that the world is a dynamic, changing, evolving phenomenon in which stability and continuity are on-going accomplishments—that take on-going work to keep in place. Thus, we do not start with stability (or stable entities with particular attributes or qualities) and then need to somehow explain change. Our presumed entities—the things we note in the world around us—do not have essential characteristics or attributes, in and of themselves (traditionally referred to as primary qualities) in isolation from other entities. Rather, entities are on-going accomplishments that become what we take them to be in relation to, or in interaction with, other entities. More forcefully stated, this ontology would suggest that there are no prior entities as such, which then interact, rather entities (in as much as they can be said to exist as such) are exactly the outcome of these interactions. They are the effects of becoming not an original source or cause.

Another way of expressing this position is in terms of the traditional notion of boundaries, which demarcates what are essential (primary) qualities as opposed to what are secondary, that is, the outcome of interaction. The ontology of becoming suggests that there are no fixed, or at least stable, boundaries that can separate the supposed primary/essential from the supposed secondary/peripheral such that any interaction will only tend to happen in the periphery of the secondary qualities and not in the centre of the primary or essential qualities. Bergson and Whitehead would argue that such an assumed boundary (between primary and secondary qualities) simply does not exist in actuality. Indeed, they would suggest that qualities as such do not pre-exist but are the emergent accomplishments of on-going becoming. As such relations in/between beings do not merely change them in their secondary qualities, rather, they are constitutive of those very beings—hence Barad's (2003) term 'intra-action.' Intra-actions are, she suggests, "ontologically primitive relations—relations without pre-existing relata."

Without the assumed ontological boundary of completeness (or of primary/secondary qualities) we cannot posit an entity that would then proceed to have relations/interactions with other beings (also already posited). Such a move or framing transforms process and becoming into entity and being which leads to an inappropriate outcome for our thinking—what Barad (2003) calls 'thingification'. Nietzsche (1996) uses the example of lightning to

illustrate this point. He suggests that when we say "the lightning flashes" we posit a pre-exiting being 'lightning' which then goes into relation with something else (the observer, for example) by 'flashing'. In the same way such a being ontology would suggest that we posit a pre-exiting human which then goes into a relation with technology by 'using' it. In contrast the ontology of becoming would suggest that through the multiplicity of incorporation of tools, gestures, speech, language, communication systems, diet, dress, roles, organisation, etc. emerged a heterogeneous assemblage we now call the modern 'human'—that is, the human (as we understand today) is an effect, or performative outcome, not an original source of all of these incorporations. Significantly, there would have been (or is) no point at which we could say that the 'human' is/was now 'complete' and everything subsequently incorporated is merely secondary—or at least any such point would have been rather arbitrary. I use the notion of 'incorporation' here in its original meaning—that is, to 'put something *into* the body or substance of something else' or to 'unite into one body'. To incorporate is not to 'add something on to an existing body' rather it is an intra-action where a new and potentially very different body exits after incorporation—this is what Haraway (1991) refers to as our becoming cyborgs .

As entities become—through intra-action, or what Pickering (1995) refers to as the 'dance of agency'—it is not possible to reduce its doing (action or agency) to any of the intra-actions that constitute it as such. Thus, in becoming/performativity there is a fundamental irreducibility operating. This has important implications for how we might think about governance. A further point to reemphasise is that, according to this ontology, when entities incorporate other 'entities' (or rather becomings) they *inherit* something from those entities (not just as secondary qualities) but in the sense of becoming new and very different entities—that is, in their supposed essential nature. A different way to express this is to say that entities (such as we humans) are not ontologically fixed but rather ontologically open and emerging. The human today is a very different human to any of its antecedents—and this is as a result of the entities we have incorporated. To summarise: through the on-going incorporation of other entities we become the beings we assume ourselves to be. Performativity produces (in intra-action) what is already assumed in interaction—and much more besides. Let now turn to the performativity of our incorporation of algorithms into our sociomaterial becoming (Introna 2011) .

The performativity of algorithms: what plagiarism detection algorithms do?

Algorithms are performative—but only as incorporated into sociomaterial assemblages, that is, through a multiplicity of intra-actions. In Donald MacKenzie's (2008) words, they are 'engines not cameras'. By this he means that algorithms do not simply express the world—the intentions of the programmers and the functions and processes that it supposedly automates. They do not 'model' the world. Rather, these algorithms performatively enact a world (produces different entities) as they become incorporated in situated action. In his work MacKenzie shows that mathematical models and financial trading systems do not simply automate the calculation of value it actively produces such value as the various

actors involved (such as markets, traders, investment managers, analysts, etc.) incorporate them into their investment practices. Moreover, such incorporation of algorithms "can have effects even if those who use them are sceptical of [their] virtues, unaware of its details, or even ignorant of its very existence" (Mackenzie 2008, 19).

In order to demonstrate that algorithms do not simply automate activities, and the actors that these activities assume, but that they performatively produce these entities (and in ways not necessary anticipated those involved), I want to discuss an example of plagiarism detection algorithms (and also mention Google). I shall do this by considering a number of questions. These questions aim to make visible the performativity of algorithms (it allows us to do a sort of a performative archaeology):

- Why did it seem appropriate for us (as part of what we do) to incorporate particular algorithms / digital entities? What did we assume that it is that they do in terms of our particular situated practices?
- What did we inherit from the doing of these algorithms / digital entities? What are inside the blackboxes and why does it matter?
- What were the performative outcomes of the incorporation and inheritance? Why does it matter?

Why is it so obvious to 10,000 institutions in 126 countries that plagiarism detection algorithms are needed as part of their educational practice? More generally stated, why is it so obvious that plagiarism detection (PD) needs to be incorporated into our learning and teaching practice? Some educators will respond by saying that there has been a dramatic increase in plagiarism in recent years. Specifically that the practice or electronic writing, and the fact that most of the sources are available in electronic form, encourages a form of writing by the 'cutting and pasting' of material into essays/assignments. Some may indicate that this is a problem because students that write by 'cutting and pasting' do not learn proper writing skills—in other words that these practices are educationally damaging. Others, the majority, would say that this is a problem because it is cheating—that is, presenting another's work as your own, hence the charge of 'plagiarism'. In other words that this is a matter of dishonesty—a lack of 'academic integrity', they might say. Most students express it slightly differently. They say that the plagiarism of other students undermine the value of their academic award. I would suggest that it is important for us to understand what is at stake in the charge of plagiarism in order to understand what PD is actually doing (rather than what we suppose it is doing) as incorporated into our educational practice. What is at stake in this discourse on plagiarism? Why is it framed in the language of cheating/ stealing?

When plagiarism is discussed the canonical reference deployed would be that of the Roman poet Martial (Terry 2007). The English word 'plagiarism' comes from Martial's appropriation of existing Roman legal concepts for his purposes—*plagiarius*, meaning 'kidnapper', or more specifically the crime of *plagium* (the kidnapping of another's slave or child). The accusation

of the crime of plagiarism by Martial was unusual in that poetry was considered, by his contemporaries, to operate in the gift economy (mostly based on patronage)—that is, in the manner in had been constituted in the oral tradition for many centuries (Ong 1982). It was also common for poets to recite each other's work (and even to appropriate elements of it in their own work)—in what White (1965) called the 'classical episteme of imitation'. Why then this claim of the crime of *plagium*? According to Mira Seo (2009) Martial differs from other poets of his time, such as Horace—who had also often complained about the inappropriate appropriation of their work—in that he introduces notions of *damage*, not in terms of a loss of poetic fame, but rather in economic and legal terms. For him the lofty immaterial world of the poetry (exchanged in a gift economy) needed to be challenged by the real economy of materiality and object exchange. Why this transformation? Most poetry at this time was 'published' as performances—often recited by a learned slave under the direction of the poet (Winsbury 2009). However, as the Roman Empire expands so too does the reach of the manuscript, thereby unsettling "the social relation of reader and writer; patronage relations based on direct contact were now felt to be incongruous with the reality of contemporary consumption" (Loewenstein 2002, 126). Thus, when poetry becomes incorporated in the manuscript as a book (becomes reproducible and mobile) and taken up by the bookseller (enters the market) it becomes produced as a commodity with economic value, that is as property. Mira Seo (2009, 583) suggests that "in Martial's poetry, the book as object is never far from the marketplace: its promiscuous distribution is associated with its status as merchandise, with an implied monetary value. And once this monetary value has been established, plagiarism can enter the literary discourse." Thus, he concludes that "plagiarism requires commodification to work" (p. 590)—in other words it needs to be incorporated as material property rather than as performance and a gift. I would suggest that if 'plagiarism requires commodification to work' then this is where we need to look to understand why their use/incorporation seems so obvious to many institutions, and importantly, what performative outcomes such incorporation produces.

The charge of plagiarism and PDS make sense in an educational market where assessments are, not about learning but rather an important element or mechanism for the commodification of education. In this market essays/assessments are exchanged for credits and credits for awards – and awards ultimately for employment (obviously in a complex and mediated manner). What is at stake for the institutions is not education and learning, in the first instance, but rather the risk to the brand value of their product – likewise for the students that participate legitimately in the exchange. In their view, held implicitly or explicitly, the students who copy and paste 'cheat' or 'steal' because they offer a counterfeit product as the real deal. And since it is a problem of cheating and theft the matter is dealt with, by the institution, as a legal issue. There are disciplinary hearings to establish the facts and students who are found guilty are expelled and often barred from getting awards from other institutions. In this, mostly implicit, discourse of commodification and exchange the plagiarism detection algorithms detect those that cheat

or steal – that is what it is supposed to do. The purpose of detection is both to detect cheating and to produce the evidence base for its prosecution.

However, if considered more carefully we have to admit that so-called plagiarism detection algorithms cannot detect plagiarism – if it is understood as presenting another's work (not merely words) as your own. This is a problem that cannot be solved computationally (at least not yet, some might say). What can be solved computationally is to detect exact copies of strings of characters in a text when compared with source texts in a reference database. This is especially significant when the reference database consist of billions of documents (the internet). Thus, in our current plagiarism detection systems plagiarism becomes computationally defined (as copied text) and by incorporating these algorithms in our teaching practice we inherit this understanding and enactment of what plagiarism is – i.e. we inherit this computational reductionism—and much more besides

Given this algorithmic enactment of what plagiarism is what did we inherit from the doing of these algorithms? What are inside these algorithms and why does it matter? Let us consider the Turnitin algorithms or system as our example. Turnitin detects similarity *when a sufficiently long string of consecutive characters from the 'original' is retained* in the electronic text. Moreover, the location, within the copied fragment, of the consecutive string of characters, is important due to the sampling window. In some cases a small amount of change in the right way (or place) will make a copied fragment undetectable and in other cases a large amount of editing will still make it possible to detect. Students who are linguistically competent can often take copied text and 'write over it' in order to render it undetectable – others with less competency struggle to do this as they often lack the vocabulary and grammar skills to achieve it . *Thus, what so-called plagiarism detection systems often detect is the difference between skilful copiers and unskilful copiers*.

What were the performative outcomes of the incorporation and inheritance? Why does it matter? Through the incorporation of these algorithms in our teaching practice we produce the skilful copier as an 'original' author (confirmed by the 'originality' report of Turnitin) and the unskilled copier as a plagiarist (a cheat and a thief). This is not to say that all those that are not detected are merely skilful copiers—indeed there are many excellent essays that rightfully get identified as not being copied. However, it is important to emphasise again that these algorithms do not detect plagiarism, only copied text. Moreover, I believe it is an important educational question of whether, and in what way, we should be engaged with copying (or imitation) completely separate from the question of plagiarism (Hayes and Introna 2005). In my view plagiarism (and what it implies) is not a valid educational question at all. Yet, by incorporating these systems into our educational practice we are reproducing the view that education (and in particular assessments) is an economic exchange (it is a market). This has many performative outcomes. For example, a clean Turnitin report confers on the text (and by implication the author) an attribute of originality. As such students start to assert the value of their commodities—such as to contest their teacher's comments and

their grades because their work is in fact original. If plagiarism (detected copying) is property theft then original work (not copied work or undetected copying) is property creation. Thus, this sociomaterial assemblage enacts the students as producers of commodities—performativity flows in both directions. As such students see it as legitimate to take their 'original' work as valuable commodities that can be sold in the market—on Ebay, for example. Moreover, if assessment is about the creation of a commodity (a 'clean' essay) then this same economic imperative would suggest that it is entirely appropriate to outsource the creation of this commodity to somebody else at a fair price. Thus, within this assumed logic we see the emergence of ghost-writing service for academic work—and we are surprised that our students turn to these for their essays?

As more and more institutions incorporate PDS into their practice this performatively produced understanding of plagiarism (and originality) is becoming taken for granted as the way the world has become, with many unexpected and implicit performative outcomes. What sort of students do we produce, in this sociomaterial nexus, which sees education as the production of commodities? What happens to education when learning becomes a matter of being detected or not? More significantly, what does this tell us about governing algorithms? It seems to me that in governing algorithms we need to become aware of the performative nature of algorithms as they become situated in sociomaterial practice (Introna and Hayes 2011; Orlikowski 2007). Also, in considering the doing of these algorithm, in these situated sociomaterial practices, we note that this doing is irreducible to any of the intra-actions. It was not the intention of the Turnitin programmers to discriminate against those with weak linguistic skills—often non-native speakers. Also, many teachers adopt these algorithms with a sincere intention to be fair. And indeed Turnitin did not create the discourse of commodification in education. However, by incorporating Turnitin into our educational practices a certain understanding of education is becoming normalised –with many irreducible performative outcomes.

There is much more to be said about the performativity of plagiarism detection algorithms—especially in the way in which they are actually incorporated in various institutions and practices in the many places where it acts. As Latour (2005, 12) suggests, we need to 'learn from them what their collective existence has become in their hands' and which 'new associations…they have been forced to establish'. As we do this we will note that in this sociomaterial nexus agency is complex matter: it is "borrowed, distributed, suggested, influenced, dominated, betrayed and translated. If an actor is said to be an actor-network, it is first of all to underline that it represents the major source of uncertainty about the origin of action…"(Latour 2005, 46).

Governing the performativity of algorithms

Being able to locate or attribute agency, in some way, is a necessary condition for governance. I would suggest that central to our normal conception of governance is the

notion of agency—what is actually done by whom, where and when. If we approach the problem of agency from a substantialist (or being) ontology then we will tend to focus on entities and the way entities interact with each other. The central assumption is that entities are in some sense the *origin* of action –- algorithms, programmers, users, and so forth. Thus, such ontology would tend to say that when we govern we need to focus on the originating acts of entities. For example, we must get Google (or Tripadvisor) to open up its ranking algorithm so that we can see what it actually does. Or, we may suggest that we need to educate the users to understand what it is that these algorithms do, when they rank, so that they can use them appropriately. These suggestions are not wrong as such—indeed, they may be useful of course. The problem, it seems to me, is that in a way they do not really capture what actually gets done in the becoming of actors in a sociomaterial assemblage, as was suggested above. They do not appreciate the problem of irreducibility as put forward by the ontology of becoming. In sociomaterial assemblages entities become what they are in their intra-action with each other. Thus, the doing of algorithms must be understood within the nexus of intra-actions in which it functions as the entities that it is becoming.

As such it would not necessarily be helpful for Tunitin to simply make public its detection algorithm. Indeed, some may argue that this will only make it easier for the cheats to cheat. Similar arguments have been made about Google's algorithm. The issue is not the details of the algorithm in isolation, as such. More important than the detail of the algorithm—in terms of establishing what algorithms actually do—is the debate of why we are incorporating this algorithm into our educational practice? Why do we conceive of copying as an act of cheating and stealing? What are the assumptions behind such view and the incorporation of algorithms within this framing of education? What if the framing was very different? Can copy detection algorithms function as an actor within an educational practice where academic writing is a process of development of the individual writing practice rather than primarily a mechanism for allocating credit. How might such an incorporation be structured or enacted? What will the algorithm do then? There is evidence of this approach where Turnitin is used as part of a formative approach to the development of academic writing (Davis and Carroll 2009).

These same arguments can also be made, for example, when we look at Google. Google's search and ranking algorithm may have started as a project to give the user the most relevant results (based on the Google index). However, today Google is not a search engine company by any stretch of the imagination. It is primarily an advertising agency (and much more besides). The developments of Googles products are not merely to give users more useful tools. Rather it is to create digital platforms to collect as much information about the user as possible in order to target advertising as effectively as possible. The information available to Google to segment the market is unrivalled in history—they may indeed achieve the elusive 'market of one' that advertisers often dream of. So individual is their segmentation that Google's Eric Schmidt's (2010) suggested that Google knows "roughly

who you are, roughly what you care about, roughly who your friends are…. Google also knows, to within a foot, where you are." As such he believes that "most people don't want Google to answer their questions, they want Google to tell them what they should be doing next." Differently stated, Google's view of what their algorithms do is that they track and profile you so that they know more about your desires and needs than you know yourself. With this individualised knowledge of you they can offer you what you need even before you realise it—they can tell you 'what to do next'. That is, offer you the search results even before you realised that you needed to search for it. That is why Google wants to combine the information it has on users across all their products—as implemented in their new privacy policy, which caused so much controversy. Let us recall that this was a company that started with a vision to give users fast and efficient access to most relevant information. Today Google search has become something very different. Moreover, what sort of 'user' have we become as the Google subject? Are we 'the user' or 'the used' in this sociomaterial nexus? What does this subject position mean for informed choice? Can we, as a Google entangled subject, say that we are informed subjects making informed choices? If Google tells me 'what to do next' what sort of questions does this raise for our notions of autonomy in the age of the Google subject?

If we want to govern Googles algorithms we need to understand what their algorithms do within the sociomaterial assemblage that it is becoming. Would it not be better for Google to be explicit at what is business model is and to give users the ability to explicitly manage their relationship with Google in a manner that is consistent with the sort of subject that they want to become in that assemblage. Some people may happily want to share their data in exchange for services. There might be many who want Google to tell them what to do next. Others might want to preserve their anonymity and buy Google's software and not go into such an implicit contract of exchange. This choice is of course not available now. However, becoming anonymous is not what Google has in mind as is clear from this excerpt Cohen and Schmidt's (2013) new book The New Digital Age:

> The basics' of online identity could also change. *Some governments* will consider it too risky to have thousands of anonymous, untraceable and unverified citizens—"hidden people"; they'll want to know who is associated with each online account, and will require verification, at a state level, in order to exert control over the virtual world. Your online identity in the future is unlikely to be a simple Facebook page; instead it will be a constellation of profiles, from every online activity, that will be verified and perhaps even regulated by the government. Imagine all of your accounts—Facebook, Twitter, Skype, Google+, Netflix, New York Times subscription—linked to an "official profile." *Within search results, information tied to verified online profiles will be ranked higher than content without such verification, which will result in most users naturally clicking on the top (verified) results. The true cost of remaining anonymous, then, might be irrelevance; even the most fascinating*

> *content, if tied to an anonymous profile, simply won't be seen because of its excessively low ranking.* (Chapter 2, emphasis added)

It is interesting how this quote starts with 'some governments' and ends with something like 'if we do not know your identity we will make you irrelevant'. Does this mean that Google is evil? No, they are merely indicating some of the performative consequences of incorporating Google into our sociomaterial practices of searching, viewing, analysing, organising, communicating, locating, and so forth. What it means to become a Google subject. The more we embody Google the more we will be expected to live a life of being located, tracked, profiled and targeted for content we are assumed to already need—that is, become the person they suppose we already are.

Some concluding thoughts

The theme of this conference suggests that algorithms are particular entities that pose particular challenges for those that govern. I believe society is particularly concerned with algorithms for a variety of good reasons, of which I would suggest the two most important one are—inscrutability and executability. There is a general view that algorithms are becoming embedded into the sociomaterial infrastructure of our lives and ordering it in more or less significant ways—and most of this is beyond our control (or even the control of those who should be in control). One response to this problem of governability is to subject these algorithms to more intense scrutiny (by those who govern)—by opening the blackboxes, as it were. My suggestion would be that this would more or less fail as it does not appreciate the performative nature of the doing of algorithms, as part of a sociomaterial assemblage. What I have hopefully shown is that what algorithms do is not 'in' the algorithms as such. This is not to say that the code in the algorithm is irrelevant. Indeed it can be very relevant—as the example of plagiarism detection has shown—but that is just one part of the story. It is just one actor in the dance of agency. Perhaps a more significant part of the story is the manner in which these algorithms—as part of a sociomaterial assemblage—condition the performative production of practices and subjects. For example, the way in which PDS are participating in the performative production of the student as a producer of property and not as a subject of learning. Or, how Google is producing the passive subject who becomes defined and performatively produced by the profile he or she happens to be an instance of.

It seems to me that if we want to govern algorithms we need to understand them within sociomaterial assemblages of becoming. We need, in my view, more multi-disciplinary research projects that can begin to understand the performative nature of our incorporation of these technologies into our situated practices. We must also however accept that any attempt to govern will itself be performative. Our attempts will become incorporated into the practices to produce what we might not have imagined—every incorporation leads to a variety of performative outcomes (and in all directions). This means that there is no 'once

and for all' intervention available to those that govern. Stated differently, governing also means governing our own attempts at governing. There are simply no simple answers available to those who want to govern. Governing is complex, on-going and imprecise.

References

Barad, Karen. 2003. "Posthumanist Performativity: Toward an Understanding of How Matter Comes to Matter." *Signs* 28 (3) (March 1): 801–831.

———. 2007. *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*. Duke University Press.

Butler, Judith. 1990. *Gender Trouble: Feminism and the Subversion of Identity*. New York: Routledge.

Cohen, Jared, and Eric Schmidt. 2013. *The New Digital Age: Reshaping the Future of People, Nations and Business*. Hachette UK.

Davis, M, and J Carroll. 2009. "Formative Feedback Within Plagiarism Education: Is There a Role for Text-matching Software?" *International Journal for Educational Integrity* 5 (2): 58–70.

Fuller, Matthew. 2008. *Software Studies: a Lexicon*. Cambridge, Mass.: MIT Press.

Galloway, Alexander R. 2004. *Protocol: How Control Exists after Decentralization*. London: MIT Press.

Graham, Stephen D.N. 2005. "Software-sorted Geographies." *Progress in Human Geography* 29 (5) (October 1): 562 –580. doi:10.1191/0309132505ph568oa.

Haraway, Donna J. 1991. *Simians, Cyborgs, and Women: The Reinvention of Nature*. 1st ed. London and New York: Routledge.

Harman, Graham. 2009. *Prince of Networks: Bruno Latour and Metaphysics*. Melbourne, Australia: re.press.

Hayes, N., and L. D Introna. 2005. "Cultural Values, Plagiarism, and Fairness: When Plagiarism Gets in the Way of Learning." *Ethics & Behavior* 15 (3): 213–231.

Hayles, N. Katherine. 2005. *My Mother Was a Computer: Digital Subjects and Literary Texts*. Chicago: University of Chicago Press.

Introna, L. D, and N. Hayes. 2011. "On Sociomaterial Imbrications: What Plagiarism Detection Systems Reveal and Why It Matters." *Information and Organization* 21 (2): 107–122.

Introna, L. D, and D. Wood. 2004. "Picturing Algorithmic Surveillance: The Politics of Facial Recognition Systems." *Surveillance and Society* 2 (2/3): 177–198.

Introna, Lucas D. 2011. "The Enframing of Code: Agency, Originality and the Plagiarist." *Theory, Culture & Society* 28 (6): 113–141. doi:10.1177/0263276409104967.

Jenkins, Holman W. 2010. "Google and the Search for the Future." *Wall Street Journal*, August 14, sec. The Weekend Interview. http://online.wsj.com/article/SB10001424052748704901104575423294099527212.html.

Kowalski, Robert. 1979. "Algorithm = Logic + Control." *Commun. ACM* 22 (7) (July): 424–436. doi:10.1145/359131.359136.

Latour, Bruno. 1988. *The Pasteurization of France*. Translated by Alan Sheridan and John Law. New edition. Cambridge, Mass.: Harvard University Press.

———. 2005. *Reassembling the Social: An Introduction to Actor-network-theory*. Oxford: Oxford University Press.
Lessig, Lawrence. 2006. *Code*. New York: Lawrence Lessig.
Loewenstein, Joseph. 2002. *Ben Jonson and Possessive Authorship*. Cambridge: Cambridge University Press.
Mackenzie, Adrian. 2006. *Cutting Code: Software and Sociality*. New York: Peter Lang.
Mackenzie, Donald. 2008. *An Engine, Not a Camera: How Financial Models Shape Markets*. Cambridge, Mass.: MIT Press.
Mira Seo, J. 2009. "Plagiarism and Poetic Identity in Martial." *American Journal of Philology* 130 (4): 567–593. doi:10.1353/ajp.0.0084.
Nietzsche, Friedrich Wilhelm. 1996. *On the Genealogy of Morals: a Polemic : by Way of Clarification and Supplement to My Last Book, Beyond Good and Evil*. Translated by Dr. Douglas Smith. Oxford: Oxford University Press.
Ong, Walter J. 1982. *Orality and Literacy: The Technologizing of the Word*. New edition. London and New York: Routledge.
Orlikowski, Wanda J. 2007. "Sociomaterial Practices: Exploring Technology at Work." *Organization Studies* 28 (9): 1435 –1448. doi:10.1177/0170840607081138.
Pickering, Andrew. 1995. *The Mangle of Practice: Time, Agency, and Science*. Chicago: University of Chicago Press.
Terry, Richard. 2007. "'Plagiarism': A Literary Concept in England to 1775." *English* 56 (214) (March 20): 1–16. doi:10.1093/english/56.214.1.
Thrift, Nigel, and Shaun French. 2002. "The Automatic Production of Space." *Transactions of the Institute of British Geographers* 27 (3) (September 1): 309–335. doi:10.1111/1475-5661.00057.
Ullman, Ellen. 1997a. *Close to the Machine: Technophilia and Its Discontents*. San Francisco: City Lights Books.
———. 1997b. "Elegance and Entropy: Interview by Scott Rosenberg." *Salon*. http://www.salon.com/21st/feature/1997/10/09interview.html.
Wendy Hui Kyong Chun. 2008. "On 'Sourcery,' or Code as Fetish." *Configurations* 16 (3): 299–324. doi:10.1353/con.0.0064.
White, Harold Ogden. 1965. *Plagiarism and Imitation During the English Renaissance*. New York, NY: Octagon Books.
Whitehead, Alfred North. 1978. *Process and Reality: An Essay in Cosmology*. New York, NY: Free Press.
Winsbury, Rex. 2009. *The Roman Book: Books, Publishing and Performance in Classical Rome*. London: Duckworth.
Wittgenstein, Ludwig. 2001. *Philosophical Investigations: The German Text, with a Revised English Translation*. Translated by Gertrude Elizabeth Margaret Anscombe. Oxford: Wiley-Blackwell.