“Don’t Ask!”: Human algorithms, tacitness, the enfant terrible, and the convention of not talking about certain things

M.B. Twidale. GSLIS University of Illinois at Urbana Champaign

This workshop is about the impacts of computer algorithms on our lives as people. However, rather like Ziewitz [2011] I think we can gain insight into this issue by looking at the ideas from a human algorithm perspective. Thinking about the hidden algorithms in computers that can and do affect us can be difficult. The algorithms may not be available, may be very complex, and those based on machine learning may have evolving results. It can be hard for less technical people to understand the underlying issues and their importance. Frankly it can be hard for those of us who consider ourselves somewhat technical.

I want to think about the hidden algorithms that we humans follow and seem to have in our heads. They also affect us too. Maybe we can gain something from examining them. From looking at the problems with human algorithms, it may be that we can uncover issues that will also play out in our analysis of complex, hidden and dynamic computer algorithms.

There are many different kinds of human algorithms and computational neuroscientists study lots of these. However I want to consider those algorithms that are amenable to conscious consideration and have a social component, since they more closely parallel certain social aspects the algorithms that are often discussed as having impacts in Google, Facebook, etc. These human social algorithms may be hidden, but they are not that mysterious; just rather awkward to talk about

By human algorithms I mean the rules that people seem to follow when they do certain things. These are like computer algorithms in their rule-like nature, but as human beings we sometimes choose to not follow rules. I don’t think it is possible for a computer to violate its algorithm – unless it is following a different algorithm that requires it to do so.

I am busy collecting examples. Here are a few. I will elaborate just a couple below. I would like to collect more. I want to have a fairly large heterogeneous set in order to see what kinds of patterns and exceptions arise.

- The redlining processes in mortgage allocation decisions in the USA
- The way people negotiate certain important sales (cars, houses, etc)
- English grammar
- Latin grammar
- Tipping practices in the USA
- How people decide where to sit on a bus
- The rules of who got to sit on special seats in the Paris Metro

I have chosen one to elaborate on: a rule of Latin grammar that has percolated into English usage in ways that some people see as rather problematic nowadays. The Latin word I’m thinking of is “alumnus”, a word we use a fair bit in universities, not least because of the importance of fundraising from alumni.

Here is the algorithm in example format, listing various inputs and the resultant outputs. It is a very simple algorithm, and so it is actually possible to be exhaustive:

- one man is an alumnus
- one woman is an alumna
- two men are alumni

- two women are alumnae
- twenty men are alumni
- twenty women are alumnae
- one man and one woman are alumni
- twenty women and one man are alumni
- twenty thousand women and one man are alumni

There are four endings:
-us Masculine Singular
-a Feminine Singular
-i Masculine Plural
-ae Feminine Plural

When there are a mixture of genders in the group, the masculine plural form is used.
I think the best way to explain this to native English speakers who are unfamiliar with the idea is as:

- See alumnus, think alumman
- See alumna think alumwoman
- See alumnae, think alumwomen
- See alumni think alummen

I would claim that the more I elaborate the algorithm, the more explicit I make it, the more sexist it seems to appear. It follows the rules of Latin grammar, but violates our evolving rules in English about avoiding sexist language.

The ACM encourages us to be careful about sexist language. For the CHI conference we are enjoined: "Be careful with the use of gender-specific pronouns (*he*, *she*) and other gendered words (*chairman*, *manpower*, *man-months*). Use inclusive language that is gender-neutral (e.g., *she or he*, *they*, *s/he*, *chair*, *staff*, *staff-hours*, *person-years*)."
I wonder what the CHI algorithm is for referring to an alumna, or a Latina alumna, or Latino alumni? Notice the squirminess that ensues when we make explicit our concerns about sexist language and if they interact with our concerns to not to be offensive to people who speak Spanish. Is English sexist? Is Spanish? Are we allowed to talk about this? Are we allowed to change the language used by other people?

This issue became more salient to me when I was chatting to a senior professor of computer science and remarked how I found the word alumni archaically sexist in the modern context. She was surprised because she did not know the rules of Latin grammar (and had just assumed it was a gender-neutral plural – well some would claim it actually is – just as they would claim 'mankind' is gender neutral). She was rather shocked when I made the algorithm visible to her, especially as how one alumnus walking into a room of 50 alumnae makes them 51 alumni.

The example is useful because the algorithm is simple, but not immediately visible to all. It also offers one the opportunity to point to certain campus buildings with "Alumni Relations" or some such over the portico and mutter smugly about "sexism written in stone". It is very satisfying to assert superior political correctness and a privileged classical education in a single aphorism. By pointing out the problem to my colleague I'm not sure I made her a happier person, but I did make the algorithm visible to her and derived not very laudable pleasure in showing off.

Riffing further on this example, and bolstered by other examples I don't have room for here, I think that there are certain aspects of these kinds of human algorithms and especially our reactions to them that are worth thinking about. They may help us in the trickier case of those complex hidden computer

algorithms. There are some things that we just do, but that we don't normally make explicit. This is a category of tacit knowledge [Schmidt 2011]. But I think that there are different kinds of tacitness in play here.

There can be rules that we follow, but don't actually know that we are following them. For native English speakers, in particular those who were never formally taught grammar at school, we may be able to speak grammatically, and reliably know when an ungrammatical sentence sounds wrong, but be unable to say why we know that. In a sense we are using our 'learn English' machine learning algorithm; it works reliably, but we don't have any access to the code or the rules – we can just effortlessly execute them. This can be exasperating for non-native speakers who have laboriously learned many rules, when they ask us about a fine point of English usage and we can easily say what is right and what is wrong, but annoyingly (for our diligent interlocutor) are incapable of saying why other than a shrug, a smile and saying "well that one just feels right".

There can be other rules that we follow, that we are at least semi-conscious of, but that we get deeply uncomfortable about if someone is sufficiently gauche as to ask us to clarify, elaborate, or one might say 'de-tacit'. Often this can occur when we have the case of the enfant terrible who insists on asking in a very loud voice why we are doing this and why we aren't doing what we do in another setting. With children this can often occur when things done at home are just not done in other public settings – ba enough when it is the child's behavior that must change, but especially mortifying when the child remarks on how certain adults are behaving differently in this different setting. I leave it as an exercise to the reader to generate their own personal examples of this widespread phenomenon.

There can also be rules where we pretend (or even believe) that the algorithm is one thing, but in actuality it is something else – typically something rather less nice. My chosen example here is tipping in the USA. For a meal, the stated algorithm asserted by many is some variation on 15% with a modification based on the assessed quality of service. I can query the sanity of this stated algorithm (should food quality play a role? Should you tip a disgusting meal served well better than a delicious meal served poorly? What counts as good service – a waiter who silently does everything before you even notice or one who keeps drawing attention to himself and interrupting to ask if everything is fine?). But the key point here is that I don't think people actually follow the algorithm that they claim to follow. It is not that they act randomly, rather that they are following a different algorithm. For example, the tip often seems to be a function of computational and monetary convenience, frequently rounded to a whole number of dollars. For modest meals this can have an enormous effect on the actual percentage value of the tip. Pointing out these inconsistencies in tipping and venturing to ask how the waiter's gender, race and physical attractiveness are factored into this determination of a significant part of a worker's income does not make for pleasant dinner conversation.

Investigating certain human algorithms is problematic. It can be that "one is just meant to know", that is, that as a member of a certain class system there are rules that we follow and we congratulate ourselves on knowing those rules and laughing at the faux pas of those who do not. In this case the hiddenness of the algorithm has value to those who know the secret and can assess who is part of the in group and who is out. Needing to ask, to reveal the algorithm is the mark of the outsider, the child, the foreigner, the enfant terrible. Asking to make human algorithms issues explicit is often an indicator of the socially inept, those who may be less good at social relationships, social rules, and indeed reading people to see that even asking the question is causing distress. People can squirm when asked about certain rules.

We have known for a long time that people often prefer ambiguity, nuance or perhaps plausible deniability. A degree of tacitness and lack of explicitness can be of value in certain human-human relationships. That can be confusing and annoying to some people, and is very problematic in our design of computational tools. In general, it is much easier to write programs if everything is explicit, clear and

preferably binary enough to be amenable to if-then rules. It is not surprising that some people who get confused by the ambiguity and tacitness of certain human-human interactions gravitate to the enforced explicitness of programming. A classic example of this human-computer tacitness preference mismatch is The Coordinator [Flores et al. 1988]. By making certain requests more explicit, this pioneering CSCW application aimed to facilitate cooperative work. But it seems that in various circumstances people just don't want to be that explicit when dealing with people.

Thinking about human algorithms seems to be rather useful for a number of reasons. It reveals a number of kinds of discomfort about making things explicit. Often this can shed light on certain not very pleasant conventions or consequences, so although unpleasant it can still be useful. Algorithmic investigation often requires systematic testing, rather like the testing of a piece of code (or its underlying algorithm). We want to know if the algorithm is reliable. Does it do what we want under a range of inputs. Indeed we really would like to be reassured that it will do what we want under all possible input conditions. To do this kind of testing requires a way of thinking that I suspect is not really natural; many people learning to program struggle with it. It requires a kind of thinking-in-parallel coupled with cussedness. By contrast people seem more inclined to think in terms of single idealizations. That is, a naïve programmer will write their code so that it takes a typical input and produces the intended output. They will test it on that typical input, maybe a few variants of it, and then declare the program to be working. Of course what they should be doing is considering all possible inputs as sets of possibilities, and focusing on the annoying pathological edge conditions. Good testers always try inputs of zero, null, gibberish, huge vales etc. One develops a skill in imagining problematic edge cases.

That is all highly desirable in computer algorithms. I think it is useful in human algorithms too, but it does not make you popular. It is why I delight in the case of the 100 alumnae being joined by an alumnus, or trying to figure out how much people would tip for a $6 sandwich, or one that was $6.25. Just as has been noted in the work on algorithm audits, the way you test a human algorithm is with multiple inputs. These inputs are typically carefully selected either to explore extreme inputs, or to determine how small changes in input can cause large changes in output. Just as the owners of computer algorithms can object to such black-box testing, so humans can often object to their algorithms being similarly tested. We see this most explicitly in politicians refusing to answer hypothetical questions. I know why they don't want to, but why do we put up with it? I want to know their decision-making algorithm, and the best way to test it is through multiple input tests (there can be valid reasons to refuse to make an algorithm explicit such as one's negotiation algorithm with Iran, or the WTO). This kind of testing can be exhausting for people and annoying for companies. With complex algorithms it might even look like a denial of service attack. But it is useful. Indeed with careful design you may not need all that many queries. Unlike a software tester, we may not need t know all possible outcomes, just that there are a number of problematic ones. For those just a few carefully designed edge cases may be enough to highlight that there is some problem with the algorithm that merits further investigation.

It seems that the approach of looking at human algorithms and even anthropomorphizing algorithm use can help in clarifying issues for researchers, and also help in explaining the findings to others who have spent less time on the matter. If we have these issues with our own human algorithms, it is hardly surprising we also have issues with computer algorithms. Making algorithms explicit is important work – but it may not make you popular.

References

Flores, F., Graves, M., Hartfield, B., & Winograd, T. (1988). Computer Systems and the Design of Organizational Interaction. ACM Transactions on Office Information Systems, 6(2), 153-172.

Schmidt, K. (2012). The Trouble with “Tacit Knowledge.” Computer Supported Cooperative Work (CSCW), 21(2-3), 163–225. http://doi.org/10.1007/s10606-012-9160-8

Ziewitz, M. (2011). How to think about an algorithm? Notes from a not quite random walk. Discussion paper for Symposium on "Knowledge Machines between Freedom and Control", 29 September 2011.