Coding or AI? Tools for Control, Surprise and Creativity

Alan F. Blackwell
Computer Laboratory
University of Cambridge
afb21@cam.ac.uk

Abstract
This essay presents an argument, accessible to non-specialists, for creative tools that are conceived as programming languages rather than as creative artificial intelligences (AI). The computational implications of creative experience are explained (again, for non-specialists) in terms of fundamentals of information theory. Using analogies to musical instruments, and drawing on recent advances in generative large language models, the paper explains the potential of aleatoric and stochastic elements in programming languages for live coding, and contrasts these with the opportunity for generative language models to be incorporated in programming tools.

1. Introduction
This paper relates to interactive systems that mix elements of programming and machine learning, including program synthesis algorithms and design strategies for mixed initiative interaction. The common theoretical basis spanning these fields, quite familiar to long-term delegates at PPIG (the Psychology of Programming Interest Group), are the decisions and actions that are taken by users on the basis of attention investment in abstraction use (Blackwell 2002a, 2002b), and the implications of notational aspects of the system as discussed in the Cognitive Dimensions of Notations framework (Green 1989, Green & Petre 1996) and subsequent variants such as Patterns of User Experience (Fincher 2002, Blackwell & Fincher 2010).

This issue is explored, in recognition of this 2022 workshop's theme "PPIG and the Muse," in relation to questions of creativity, and illustrated with some simple musical examples. To the extent that machine learning research continues the long-term agenda of AI, this combination of machine learning and creative programming engages directly with the longstanding concern of whether an AI system can or cannot be creative. That topic has been discussed at great length, in particular through the work of Margaret Boden (1998, 2010), and in many collections and scholarly contributions that draw on and extend her framework.

In this paper, I do not present any ideas that are fundamentally different to Boden's analysis, but I emphasise a perspective from programming, with an information theoretic account of creative experience that is intended to engage with the literature on end-user programming (where many users might have programming-like experiences that would not necessarily be described as programming languages in the conventional understanding – Blackwell 2002a, Blackwell, Robinson et al 2002). The specific genre of programming languages to which this analysis is especially relevant is live coding languages, and in particular those used for coding music (Collins et al 2003, Blackwell & Collins 2005, Blackwell et al 2023 in press). In relation to other research on end-user programming languages, this paper is more relevant to those kinds of end-user programmers who are motivated by creative and artistic questions rather than practical ones (Aghaee et al 2015, Blackwell 2017).

2. Programming Languages vs. AI design strategies
A central concern of the Psychology of Programming field over many years has been about giving users greater control, so that they are able to explain to computers what the user actually wants to do. In contrast to the field of natural language processing, which aims to have computers understand the way people *naturally* talk to other people, psychology of programming considers the alternative that an *artificial* language (or perhaps 'un-natural language') might be a more effective design approach in the longer term.

The more familiar research agenda of natural language processing can be described in terms of the Turing Test, in which a computer interacting via a typewriter keyboard would become indistinguishable

from a human (Turing 1950). However there are two ways to win the Turing Test. The hard way is to make computers more and more intelligent, until computers and humans are as smart as each other. The easy way is to construct software systems, businesses, and social networks that cause humans to act more and more stupidly, until computers and humans are as stupid as each other.

Psychology of programming therefore provides necessary fightback against the companies and organisations that push "AI" algorithms onto us, perhaps because they hope that making their customers and citizens more stupid might result in better business and less inconvenience (for the company). Critical analysis of machine learning algorithms from an HCI perspective (e.g. Blackwell 2015, 2019) has suggested some of the problems that result, and recent advances in Human-Centred AI (e.g. Shneiderman 2022) argue that instead of creating artificial intelligences, what we need to create is intelligent super-tools. Such super-tools, if they are to be created, can certainly benefit from the insights of psychology of programming that were discussed in the introduction.

3. Creativity and AI

Despite the familiar human-centric design emphasis on giving users greater control (and at PPIG, the possibility of more powerful and general control through programming), this paper considers the (relatively rare) occasions when we enjoy having a machine doing something unexpected. That is, the machine does something beyond, or other than, what the user had specifically asked for. This is related to the questions that AI philosophers ask about whether a machine can be creative, although I am going to argue that "creativity" is the wrong word, both philosophically and technically. What I'm actually going to talk about is when a machine does something *surprising*.

This distinction follows the well-established writing of Boden on the question of surprise (2010). As Boden analyses in far greater detail, there is an everyday relationship between creativity and surprise. For example when we see somebody make a creative solution to a problem, this seems creative precisely because we hadn't thought of that solution ourselves. If we had already thought of the same solution, it wouldn't be surprising, and would seem less creative as a result, despite the fact that the inventor might have engaged in exactly the same cognitive process, whether or not anybody was watching.

Much recent philosophical discussion about whether machines can be creative has focussed on analysis of a single incident – move 37 of the Go match between AlphaGo and Lee Sedol, a move which it seems none of the professional Go players or commentators had considered before. This single move has been described as if it introduced the dawn of a new age of machine creativity (Curran et al 2020). But in this argument of this paper, I'm going to argue that move 37, although clearly surprising to those directly involved, was not really very creative in the sense that humans find interesting.

4. Information Theory and Aesthetic Experience

To understand this important distinction, we need to use the principles of information theory developed as a theory of communication by Claude Shannon at the telephone company research institute Bell Labs (Shannon 1948, Shannon & Weaver 1949). In case this paper is read by non-specialists, I will note that information theory is the fundamental operating principle of all information and communication technologies. Shannon's discovery and formalisation of information theory is perhaps the most significant practical advance in mathematical physics since Isaac Newton's laws of gravity. Information theory is not yet taught in high schools, but certainly will be before long. And as far as Go players, or philosophers of creativity, are concerned, I will suggest that trying to explain a creative move without using the mathematics of information theory is like attempting a philosophical explanation of why balls move the way that they do on a pool table, but without using Newton's equations.

In its simplest form, information theory gives us a quantitative measure of the amount of information that has been sent over some communication channel. Although this seems straightforward, there are some subtle points, so subtle that they might even seem paradoxical. After more than 70 years since Shannon first started to define principles of digital communication over telephone lines, we have all become very familiar with measuring amounts of data in terms of megabits or gigabits needed to stream a movie or email a photograph. The paradoxical aspect for many is that not all *data* counts as *information* in the theoretical sense developed by Shannon. If you were to send me an email message, and then immediately send the same email message again, the second message might use a lot of data,

but it hasn't given me any more information – it is *redundant* in the technical jargon of information theory. On the other hand. if you sent the second email to a different person, it's not redundant, because *that person* hasn't seen it before. Although it is easy to measure the number of bits being transmitted along a cable, the information content is not necessarily easy to measure objectively, because it depends on the person who receives the message. If the receiver already knows what the message will contain, no information has been transferred. Information theory is a measure of that amount of information provided by a message, that the receiver does not already know. Information theory is a measure of surprise![1].

I've followed the argument of Boden and others that what we call "creativity", when done by a machine, is more precisely a measure of how much we are surprised by what the machine does. So information theory can be used as a measure of creativity. When a Go-playing robot is being observed by an expert human Go-player, and every move is exactly the one expected, there are no surprises, no subjective impression of creativity, and no information. Another person, with less expertise in Go, might be surprised by every move they see – but this is not because the robot has become more creative, just that we have found an audience who is easily impressed. In many fields of human creativity, we call an ignorant person who is easily impressed *undiscriminating*, and an expert *discriminating*. In fact, this is another technical term in information theory, and perhaps a mathematical formalisation of the proverbial understanding that creativity, like beauty, is in the eye of the beholder.

We see this all the time in human art forms, and especially in music, which is probably the easiest of art forms to describe mathematically. Some sequences of notes are very predictable. Perhaps the most predictable is when I have a favourite song, and listen to the exact same recording many times. Once I get to know it, it can be comforting to listen to a piece of music that I know really well, although also a little boring to listen to it over and over again. For most people listening to music, there is a sweet spot between things that are comforting, familiar (or possibly boring), and things that are interesting and surprising. For example, I like to listen to blues music, which often follows a 12-bar pattern, with each chord determined by its place in that sequence. Because the sequence of chords is so familiar, it's satisfying to listen to. The chords themselves, and the tune of the vocal part, are likely to follow one of the familiar patterns of Western music, such as the minor scale. For someone who listens to a lot of blues, it is possible to hum along and make a good guess at what note will come next, or even improvise your own new part to follow the rules you have learnt.

If you aren't a musician, but have access to a piano keyboard, you can get an understanding of this with a simple experiment. Just play up and down the white keys, playing one note at a time, and changing direction whenever you feel like it. You are now playing a tune in the key of C Major. After a minute or so, press one black key. You will hear this as a kind of surprise, different from the comforting (and possibly boring) routine of the C Major tune. Pretty much all Western music is built on managing the amount of surprise, including just the right number of black notes, and at appropriately surprising moments, to maintain the interest of the listener. In contrast, if you play any old combination of white and black notes at random, the result will not sound very pleasant, or even like a coherent tune. In the mid 20th-century, "serialist" composers set themselves the challenge to use all 12 white and black notes before repeating any of them. The result has not caught on, and doesn't seem to have become the basis of any familiar pop songs or comforting lullabies.

Great musical performers, in blues or any other genre, follow the rules of that genre, but also add a little surprising spice by using a different note from time to time. Pop music has its own set of rules, and every pop song sounds more or less like another song, because that's what makes pop music comforting and familiar. If you listen to heavy metal, there is a different set of conventions, and we each learn to like what we like. Heavy metal fans like stuff that might not seem appropriate to play in a restaurant, and think they have extreme tastes, but a soprano coming out to sing *O mio babbino caro* in the middle

[1] That exclamation point is not really very surprising, or creative – perhaps another example of redundancy.

of a concert by the notorious death metal band *Vile Demons of Excrement*[2] would be even more surprising than a blood-curdling scream.

The lesson from this is that we like certain kinds of surprise, but not others, even within art forms that apparently invite disruptive alternatives to mainstream culture. This points to another paradox of information theory, which is that the greatest amount of information you can send over a communication channel is a completely random sequence of numbers. Imagine a message that is composed of zeros and ones, where every bit is chosen by flipping a coin. The person receiving this message would have no way to predict what the next bit is going to be, meaning that every bit that arrives is a complete surprise. This message is as surprising as it could possibly be, but we don't perceive it as being creative. On the contrary, a sequence of random bits is described as *noise* (another technical term in information theory).

A theory of creativity in machines must make a distinction between the information theory categories of *signal* and *noise*. The signal is the message we want to hear (a meaningfully creative novelty), and the noise is random stuff that is not interesting. We are interested in messages that relate to our expectations, and to things that are familiar, whether Go, 12-bar blues, or photographs of our children. When someone jumps in with something completely random, for example noise corrupting a digital photograph, it is annoying and upsetting, not an act of creativity on the part of the camera. We do like to receive surprising messages, for example when an old friend contacts us out of the blue. But we don't like very "surprising" messages such as a loud burst of random static. The kinds of surprise that we recognise as being creative depend on two things – somebody who wants to tell us something, and our expectation of what we might hear.

5. Machine learning and surprise

Now, let's consider how these principles of information theory, which we perceive as expectation and surprise in aesthetic experiences, relate to creativity in machines.

Machine learning systems, by definition, learn to replay what they have seen before. At the simplest level, predictive text of the kind we have on our phones has been trained with a *language model* – a dictionary, and perhaps a few hints about what common word might come next. This model is used to predict what comes next according to a principle of information theory – the principle of least surprise. The word that your phone predicts is always the least surprising word, that is, the one that is most expected to come after the letters you have just entered. According to information theory as discussed in the previous section, the least surprising word is also the least creative. Indeed, who would want a more creative predictive text algorithm on their phone? A *creative* "AI" phone, which surprised you with words completely different to what you were trying to say, would be a pain in the arse.

The latest generation of large language models, based on deep neural networks that can predict whole sentences or paragraphs at a time, follow exactly the same information theoretic principle. They have been trained with far more than a simple dictionary – including wikipedia pages, online forums, and whatever other text their developers can find online for free (Bender et al 2021). Sometimes the results can be surprising, but only to people who haven't spent much time on the internet. It's somewhat surprising, but probably more horrifying, when these models produce large quantities of offensive, racist and violent text. In fact it's not that surprising that the internet is full of violent and racist text, because we seem to have designed it to be that way (Monteiro 2019).

This is a sad fact of modern life, and certainly suggests some precautions we might take before creating more powerful predictive text systems for our phones. However, none of this is evidence that the machine is being creative, and none of the resulting text is surprising in the information theory sense. It is just a communication channel, where the things that come out are the product of the things we put in (GIGO, or garbage in, garbage out, according to the old software developer's saying).

Just like music, surprises come from the random bits of noise that we didn't expect because they didn't follow the pattern. We could train a neural network language model with lots of beautiful poetry, instead

[2] Not a real death metal band.

of forum posts by violent racists, and that network would produce something that was (hopefully) more poetic. But this result would not be surprising, and it still wouldn't be creative. According to information theory, creative surprise needs to involve some kind of random decision, if it is going to be genuinely unexpected by anybody.

This is exactly how the generative language models work. They have a built-in random number generator, that produces less expected outcomes by essentially tossing a coin, as if your predictive text system occasionally threw a completely random word into one of your SMS messages, just for fun. On your phone, most of the suggestions it gives are actual words, rather than random letters, and in a large language model, most of the suggestions are actual sentences. It just that as the randomness "temperature" is turned up, those sentences become more surprising, further away from the more likely things that any person would say next.

6. Surprise as a creative tool

I've experimented with large language models trained to plagiarise my own work, by feeding 10 years of my academic papers into a neural network. If the temperature is very low, the results aren't very surprising. In fact, they are very likely to produce outright plagiarism, just repeating the kinds of sentences and phrases that I've often used myself (indeed, just as I have done in writing this paper for PPIG 2022). As the temperature is turned up, more random stuff can happen, and it becomes more entertaining to look at the results. These sentences do look kind of like things that I would write, but often with weird twists or gaps in logic. Sometimes, these even seem like creative opportunities – saying things that I've talked about before, but suddenly combined in a new way that I haven't thought of (Alexander et al 2050/2021).

This process – using coin tosses or throws of a dice to generate new ideas - is a common strategy used by music composers and other artists when they would like to develop their ideas in a new and unexpected direction. 20th century composers like John Cage were famous for "aleatoric" compositions starting from a random process (Leong et al 2006). Brian Eno's "oblique strategies" (Eno & Schmidt 1975) include all kinds of disruptive suggestions, intended to help a creative artist break out of their comfortable habits of thought and try something completely new. Painters and sculptors also like to surprise themselves through conversations with their material, where random splashes, unexpected turns in the wood grain, or slips of the chisel might help them to explore possible ideas beyond their conscious intentions.

We need to be very clear about the difference between the use of randomness as a compositional strategy, and the presentation of randomness as an artwork in itself. I've already explained that the most surprising message (in an information theory sense) is a sequence of completely random numbers or coin tosses, where there is no way to predict any number from the ones that came before. Completely random messages are very surprising, but also very uninteresting, because they communicate nothing at all. There is no hidden message in a series of coin tosses or dice throws, no matter how much we might want to find one. And of course, the human desire to find meaning has resulted in many superstitious practices where people do look for meaning in coin tosses, dice throws, or cards drawn from a deck. Tarot readings, gambling, and other entertaining performances, just like the compositions of John Cage, use random information as a starting point for human creativity.

But what we need to remember is that random information is not communicating anything. Random information is perceived as surprising precisely because there is no actual message that could have been anticipated. We can enjoy the performance of a Tarot reader, but the idea that random events have meaning in themselves is nonsense. We also need to understand that the same is true of the random processes that cause us to attribute "creativity" to AI systems. An AI system can produce surprising output if it includes random elements. But this is not creative in the human sense, because it is not a message from anywhere. In general, random elements within a digital sequence are not signal, but noise. In terms of philosophy of mind, a random message has no meaning because there is no *intention* behind it. All of these things can be directly described in terms of information theory. Philosophical speculations about creativity, when applied to surprising outputs of a randomising system, are no better than mystical explanations of why there might be meaning hidden in a Tarot card deal.

These are the simple reasons why digital machines may be surprising, but not creative. As I said earlier, Shannon's principles of information theory are as fundamental as Newton's principles of gravity, so searching for a magical basis of creativity in AI that would be contradicted by information theory is as productive as research into anti-gravity machines. Possibly entertaining in science fiction or theatrical magic performances, but not a serious basis for science or engineering.

None of this is to say that randomness is not a valuable aid to human creativity. Just as with John Cage's aleatoric composition methods, it can be exciting to experience art works with the right mix of comfort and surprise. We don't like music (or any other artwork) to be too comforting, because it is just boring. Small amounts of surprise, at the right time, add spice to our mental worlds. Sometimes, a random event from our own computer, or even an unexpected suggestion from a predictive text keyboard, might be a happy accident that we enjoy responding to, and perhaps even repeating to our friends, just as when John Cage saw a dice throw that he liked, and included that note in his composition.

In future, I expect that I will use more powerful generative language models to save myself typing. The ones I use today can already complete full sentences. Usually in a very boring way, because these sentences reflect the most expected, least surprising, thing I could say. That's OK, because I do quite often need to write a lot of boring and repetitive text, for example in this paper. I expect that I will also enjoy turning up the temperature, for more surprising randomness, on a day when I've bored myself, and trying to think of something new to say.

These kinds of tools are all great, and I look forward to using them, but we can't confuse the dice throw inside the neural network with the creative decision that I make myself when I decide just when to throw the dice, or when I choose which one of the various random suggestions is most interesting to use. At that point, *I'm the one being creative*, because I have a message that I want to send - in this essay, a message to you, the reader. If you were to throw a dice to generate a sequence of random words and letters instead of reading this paper, or if you were to read the randomised output from a chatbot, you might find it an entertaining game, but there is literally nobody sending you a message, any more than if you chose words from a dictionary by throwing a dice.

7. How to design surprising tools

To bring this back to the PPIG theme, I now turn to the kinds of programming languages that artists use. Because artists appreciate opportunities to incorporate random surprises in their work, they are unusually interested in programming languages that sometimes behave in random ways. In other circumstances, this is *not* what we want from a programming language. Certainly not a program that is managing the cruise control on a car, or the thermostat on an oven, or a nuclear power station.

In the hands of an expert user, slightly unpredictable tools can be surprisingly valuable. Aeronautics engineer Walter Vincenti (later director of the Stanford university program in Science, Technology and Society), described the long-term research effort to create more stable and predictable airplanes, which although technically successful, resulted in planes that pilots hated flying (Vincenti 1990). It turned out that an expert pilot needs a plane that is just slightly unstable, just as a Formula 1 driver performs best in a car that is always on the edge of going out of control (which would be a car that a normal person is likely to crash within seconds).

I spent several years working with a team of expert violin researchers, analysing the acoustic vibration and auditory perception of that surprisingly complex object (Fritz et al 2012). It turns out that violins, like aircraft and racing cars, are designed to be unpredictable. The mode of vibration for a string being driven by a rosined bow has an incredibly complex variety of frequency components, amplified by a wood body that is designed to vibrate in many different directions at the same time. The resulting rich timbre can be controlled by an expert player to create a huge range of different sounds. In the hands of a beginner, the surprising range of random noises that can come out of a violin make it one of the more unpleasant choices for a child's instrument. It is possible to "tweak" a string instrument to emphasise some kinds of vibration more than others, and our research team interviewed a specialist adjuster who makes a living just by moving around small internal components of the instruments played by top-flight cellists. He showed me how he could adjust a cello to make a more reliably beautiful and comforting sound, which sounded good even in my bass-player's hands (Blackwell 2022). But as he explained, this

adjustment, however much I liked it myself, is the opposite of what an expert player wants. Professional music is only interesting with that element of surprise, and it has to be available at the times the player needs it. And to return briefly to the argument earlier in this paper, it is not the violin being creative here – the random elements of surprise are a creative resource, no more than a tool for the human artist.

So how does this relate to computer-based tools? Just as with violin players and airplane pilots, professional computer artists appreciate tools that have the capacity to surprise them. I have spent a lot of time working in the musical genre of "live coding" (Collins et al 2003, Blackwell & Collins 2005, Blackwell, McLean et al 2014, Blackwell, Cocker et al 2023 in press), where music is synthesised by an algorithm that the performer creates on stage, writing the code in front of an audience. In its most popular incarnation, this is the "algorave" (Collins and McLean 2014), where a nightclub full of people may be dancing to the algo-rhythms.

Live coder Sam Aaron was working in our research group when he created his popular Sonic Pi language (Aaron et al 2013, 2014)[3]. When Sam and I were discussing the features that would be needed in Sonic Pi, it was clear that some kind of randomising function would be needed, since it is often useful in performance. At a micro-level, a small amount of randomness can make a repeated drumbeat sound less robotic, or the frequency components of a synthesised sound richer and more complex in the same way as for a bowed instrument like a violin. Random walks can also be used to choose the notes of a tune, just as in the experiment I suggested earlier where a non-musician might wander backwards and forwards over the white notes of a piano. In Sonic Pi, a straightforward approach to creating a tune-generating program might be to generate the scale of C-major, and then proceed up and down that scale at random. To make the tune slightly more interesting, the program might occasionally jump two notes rather than one, or add the occasional black note, but do those things at unexpected moments, which would be determined by another random variable.

Sonic Pi programmers certainly do these things, and most live coding performers use similar strategies, making use of the "random" keyword that is in almost all programming languages. Careful choice of the amount of randomness, just like choosing the temperature in more creative uses of a large language model, can be a source of interesting inspiration to the human artist.

However, Sam noticed an interesting thing after a year or two performing and composing with Sonic Pi, and talking to the other musicians and school students who it was designed for. Although random surprising notes can be interesting in moderation, these random tunes pretty quickly get boring, even if they are following the rules of western harmony scales and chords, because there is no artistic intention. Even worse, a randomising program would occasionally produce something that sounded really beautiful, but would then vanish, never to be heard again. Sam therefore changed the random function in Sonic Pi so that it is not really random at all. Most programming languages have very sophisticated random number generators, to guarantee (for example) that there is no way a code-breaker would be able to predict the random key to an encrypted message. The new Sonic Pi randomising function is an artistic tool that will produce an unexpected result when first used, but then do the same thing again if you ask for it. All music plays with the expectation systems of the human brain, leading us to expect one thing, surprising us with something else, then perhaps comforting the listener by repeating the same thing again. The verse and chorus in pop songs, the repeated passages in a Vivaldi concerto, and the thematic motifs of a Mahler symphony all draw us in by repeating variations of an intriguing idea until it becomes familiar.

These are also the ways that we can have creative experiences with AI systems. Surprise plays an important part in aesthetic experience, and tools that sometimes behave randomly can produce surprises in a way that is described by information theory. Tools for expert creative artists do need these kinds of capability, but the tool itself has nothing to communicate, because a random event is not a message from anywhere. AI systems are not creative, but if we can program them to manipulate comfort and surprise, they certainly become creative tools.

[3] If any readers would like to try this for themselves, just download the Sonic Pi package (free for Windows, Mac, Linux and Raspberry Pi), and experiment with its built-in tutorials and examples to make your own music synthesis code.

Insights from psychology of programming are desperately needed, in the study of machine learning in art, literature and music. It is clear from the discussion in this paper that creativity will play a part, and that surprise will play a part. As the broader population becomes more familiar with the implications of generative neural network architectures, for example through the recent popularity of the DALL-E Mini project, it seems likely that these will become more widely used as an element of creative process. The selection of suitable prompts to generate interesting output from such text-to-image systems is becoming known as *prompt engineering*, or even *prompt programming*, suggesting that insights from psychology of programming and end-user software engineering may be useful. There are certainly immediate opportunities for insight from the application of Cognitive Dimensions in this domain, since the language interfaces currently being used are woefully inadequate as a programming notation, even for creative and exploratory design activities, on many relevant dimensions.

8. Creating novel source code

Does this essay have any implications for the use of generative networks to produce program source code, including well known demonstrations such as Copilot Labs from GitHub? We might certainly imagine that aleatoric processes for source code synthesis could be a component of an artistic project or programme, perhaps alongside other practices of programming identified by Bergstrom (2016). There have also been interesting experiments, over many years, in the random synthesis and perturbation of source code to generate useful programs automatically, for example using genetic algorithms to stochastically explore the space of alternative programs that might produce a desired output. But it is widely expected that systems like Copilot would have more practical utility – "AI pair programmer," as current promotional material from GitHub suggests. The empirical literature on pair programming, for example as presented at PPIG in the past, suggests that this is not a plausible model of how such tools might work.

Nevertheless, there is certainly opportunity to improve the language models and prediction performance of the code completion algorithms currently used in leading programming editors. Anticipating welcome modes of interaction in code completion might benefit more from consideration of natural language text completion interfaces (otherwise known as predictive text) than by making unlikely analogies to pair programming. Advances along those lines should become useful and welcome, but the generative model in which natural language text input is used to generate source code output more closely resembles a stochastic compiler. If such a "compiler" from natural language to source code interprets the natural language prompt without original "creative" (i.e. random) information, then the output will be a kind of pastiche or plagiarism of the source code that the model was trained on. The distinction between pastiche and plagiarism, in this whole generation of models, is really an outcome of the temperature setting – the amount of random variation. Aside from artistic applications, it seems unlikely that many programmers will be interested in random variations on source code that would otherwise have been expected to work. As a result, the stochastic generation temperature is likely to be kept low, and such programmer assistance algorithms will therefore be closer to plagiarism (or "software reuse", as it is known when done with permission). Rather than a translation interface, the input prompt in this case is not actually a specification, but rather a query for retrieval of an appropriate piece of code (hopefully) known to work in the past.

Shneiderman (2022) makes the case that any predictive text entry interface can also be regarded as a recommender system, where the recommendation being offered relates to opportunities for time saving, improved textual "productivity" (for those paid by the word), or for improved attention savings, in the analytic terms of the attention investment framework. This perspective is likely to offer a helpful approach to the design of programming assistants that incorporate generative models. Ideally, such models will interact on a mixed initiative basis, for most effective augmentation of the human programmer, taking account of attention investment-based design strategies such as Wilson and Burnett's Surprise-Explain-Reward (Wilson, Burnett et al 2003), and Horvitz's advice for mixed-initiative interaction (1999). As with programming by example, the notational choices for explaining the proposed additions to the code are likely to be critical, as few programmers will want to accept automatically generated code whose function they do not understand (Blackwell 2001).

Overall, there is a bright future for the use of machine learning techniques to enhance programming, both for creative purposes and more mundane data processing automation. Nevertheless, meaningful input, if we avoid the temptation toward large-scale institutional plagiarism, will come from the programmers.

9. References

Aaron, S. and Blackwell, A.F. (2013). From Sonic Pi to Overtone: Creative musical experiences with domain-specific and functional languages. *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, pp. 35-46.

Aaron, S., Blackwell, A.F. and Burnard, P. (2016). The development of Sonic Pi and its use in educational partnerships: co-creating pedagogies for learning computer programming. *Journal of Music, Technology and Education* 9(1), 75-94.

Aghaee, S., Blackwell, A.F., Kosinski, M. and Stillwell, D. (2015). Personality and Intrinsic Motivational Factors in End-User Programming. *Proceedings of IEEE Symposium on Visual Languages and Human Centric Computing (VL/HCC 2015),* pp. 29-36.

Alexander, A., Bassett, C., Blackwell, A. and Walton, J. (2050/2021). *Ghosts, Robots, Automatic Writing: an AI Level Study Guide.* Cambridge Digital Humanities: Cambridge, PREA.

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?🦜. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610-623).

Bergstrom, I. and Blackwell, A.F. (2016). The Practices of Programming. In *Proceedings of IEEE Visual Languages and Human-Centric Computing (VL/HCC 2016),* pp. 190-198.

Blackwell, A.F., Cocker, E., Cox, G., McLean, A. and Magnusson, T. (2023, in press). *Live Coding: A user's manual*. MIT Press

Blackwell, A. and Collins, N. (2005). The programming language as a musical instrument. In *Proceedings of PPIG 2005*, pp. 120-130.

Blackwell, A.F. and Fincher, S. (2010). PUX: Patterns of User Experience. *interactions* 17(2), 27-31.

Blackwell, A.F., McLean, A., Noble, J. and Rohrhuber, J. (2014). Collaboration and learning through live coding. *Dagstuhl Reports* 3(9), 130-168. Edited in cooperation with Jochen Arne Otto.

Blackwell, A.F., Robinson, P., Roast, C, and Green, T.R.G. (2002). Cognitive models of programming-like activity. Proceedings of CHI'02, 910-911.

Blackwell, A.F. (2001). SWYN: A Visual Representation for Regular Expressions. In H. Lieberman (Ed.), *Your wish is my command: Giving users the power to instruct their software*. Morgan Kauffman , pp. 245-270.

Blackwell, A.F. (2002a). What is programming? In *Proceedings of PPIG 2002*, pp. 204-218.

Blackwell, A.F. (2002b). First steps in programming: A rationale for Attention Investment models. In *Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments*, pp. 2-10.

Blackwell, A.F. (2015). Interacting with an inferred world: The challenge of machine learning for humane computer interaction. In *Proceedings of Critical Alternatives: The 5th Decennial Aarhus Conference*, pp. 169-180.

Blackwell, A.F. (2017). End-user developers - what are they like? In F. Paternò and V. Wulf (Eds). New Perspectives in End-User Development. Springer, pp. 121-135.

Blackwell, A.F. (2019). Objective Functions: (In)humanity and Inequity in Artificial Intelligence. *HAU: Journal of Ethnographic Theory* 9(1), 137-146.

Blackwell, A.F. (2022). Too Cool to Boogie: Craft, culture and critique in computing. In J. Impett (Ed.) *Sound Work: Composition as Critical Technical Practice*. Leuven University Press / Orpheus Institute, pp. 15-33.

Boden, M. A. (1998). Creativity and artificial intelligence. *Artificial intelligence*, 103(1-2), 347-356.

Boden, M. A. (2010). *Creativity and art: Three roads to surprise*. Oxford University Press.

Collins, N., McLean, A., Rohrhuber, J., and Ward, A. (2003). Live coding techniques for laptop performance. *Organised Sound*, 8(3), 321-330.

Collins, N., & McLean, A. (2014). Algorave: Live performance of algorithmic electronic dance music. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 355-358.

Curran, N.M., Sun, J. & Hong, J.W. (2020). Anthropomorphizing AlphaGo: a content analysis of the framing of Google DeepMind's AlphaGo in the Chinese and American press. *AI & Society* 35, 727–735. https://doi.org/10.1007/s00146-019-00908-9

Eno, B., & Schmidt, P. (1975). *Oblique strategies.* Opal. (Limited edition, boxed set of cards.).

Fincher, S. (2002). Patterns for HCI and Cognitive Dimensions: two halves of the same story? In *Proc. 14th Workshop of the Psychology of Programming Interest Group (PPIG 14),* pp.156-172.

Fritz, C., Blackwell, A.F., Cross, I., Woodhouse, J. and Moore, B. (2012). Exploring violin sound quality: Investigating English timbre descriptors and correlating resynthesized acoustical modifications with perceptual properties. *Journal of the Acoustical Society of America* 131(1), 783-94.

Green, T. R. G. (1989). Cognitive Dimensions of Notations. In L. M. E. A. Sutcliffe (Ed.), *People and Computers V*. Cambridge: Cambridge University Press.

Green, T. R. G., & Petre, M. (1996). Usability Analysis of Visual Programming Environments: a 'cognitive dimensions' framework. *Journal of Visual Languages and Computing*, 7, 131-174.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 159-166.

Leong, T.W., Vetere, F. and Howard, S. (2006). Randomness as a resource for design. In *Proceedings of the 6th conference on Designing Interactive systems (DIS '06*, pp. 132–139.

Monteiro, M. (2019). *Ruined by Design: How Designers Destroyed the World, and What We Can Do to Fix It*. Mule Books.

Shannon, C.E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*. 27 (3), 379–423.

Shannon, C.E. and Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press.

Shneiderman, B. (2022). *Human-Centered AI*. Oxford University Press.

Turing, A.M. (1950). Computing Machinery and Intelligence, *Mind* 59(236), 433–460, https://doi.org/10.1093/mind/LIX.236.433

Vincenti, W. G. (1990). *What engineers know and how they know it*. Baltimore: Johns Hopkins University Press.

Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., Durham, M. and Rothermel, G. (2003). Harnessing curiosity to increase correctness in end-user programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems,* pp. 305-312.