Seven Catchy Phrases for Computational Creativity Research

A Position Paper

Simon Colton
Computational Creativity Group
Department of Computing, Imperial College, London

http://www.doc.ic.ac.uk/ccg

I understand that simulating creative processes by computer can enhance our understanding of creativity in humans. I also understand that there is more need than ever for software to help people to be more efficient in creative jobs. And I know that computational creativity research can be of great value in both these areas. However, I'm really only interested in the intellectual challenge of enabling nuts and bolts machines - bits and bytes computers - to create artefacts of real cultural value to society. Such behaviour used to be thought of as divinely inspired, no less than a gift from the Gods. This is why it is a worthy challenge for me to bet my career against. Building a truly computationally creative machine is as much a societal as a technical challenge, and it will need computational creativity researchers to come together in consensus about certain aspects of their field. To this end, I have written here seven phrases around which we could rally (or about which we could debate - which may also be healthy). I present the ideas from which the phrases emerged with little argumentation, in the tradition of a position paper. They are drawn from twelve years of immersion in the field of computational creativity during which I've written an automated mathematician (HR) and an automated painter (The Painting Fool), and they have created artefacts which I believe are of real value to society.

Ever decreasing circles

Let's start with the observation that it is much easier to put together artificially intelligent systems if we have something concrete to work towards, especially when there is a general and workable theory of human intelligence to guide us in. This has led to a somewhat unspoken notion in computational creativity that we should be looking towards research in natural creativity for guidance on how to get computers to behave creatively. However, while natural creativity research influences computational creativity research somewhat, our research into building creative software certainly influences our understanding of creativity in general. So, we shouldn't wait for philosophers, psychologist, cognitive scientists or anyone else to give us a workable impression of what creativity is. We should embrace the fact that we are actually undertaking research into creativity in general, not just computer creativity. So, we should continue building software which undertakes creative tasks, we should study these systems and we should help to understand creativity. In this way, there will be ever-decreasing circles of research where we influence research on natural creativity, then it influences us, and so on until we pinpoint and understand the main issues of creativity in both artificial and natural forms.

Paradigms lost

The problem solving paradigm is crippling AI research. You know the routine: an intelligent task needs to be automated, and we immediately ask the same questions: does it involve proving something; does it involve generalising a pattern; does it involve putting together a plan, etc. If it is possible to answer yes to any of these questions, then the task is pigeonholed forever as a theorem proving problem, or a machine learning problem, or a planning problem, etc. And the task is lost forever - only researchers in the designated area will work on automating approaches to that particular intelligent task. We should instead remind people that by breaking down certain intelligent tasks into sub-problems, much of the essence of the task is lost. As humans, we don't solve the problem of writing a sonata, or painting a picture. Rather, we keep in mind the whole picture throughout, and while we surely solve problems along the way, problem solving is not our goal - creating artefacts of cultural value is the purpose of the exercise. We need to resurrect the lost paradigm of artefact generation, and we should educate the next generation of AI researchers in the need to embrace entire intelligent tasks which lead to the production of beautiful, interesting and valuable artefacts.

The whole is more than a sum of the parts

It has been my observation that the more interesting pieces of software which undertake creative tasks are those where multiple systems have been combined. Certainly, the only systems I've built which I was prepared to call creative involved at least two pieces of AI software written for completely different tasks, but brought together so that the whole was more than a sum of the parts. However, there is still a tendency to re-implement techniques to fit into the workflow of a creative system rather than investigating how other people's AI software could be incorporated. For instance, I've often seen sub-tasks within creative systems being achieved - badly - with a bespoke system for generalisation that could be solved with an off-the-shelf machine learning system. Or, similarly, some deductive task is performed using a forward-chaining approach that would be laughed at by automated reasoning researchers. We should assume that anyone who has built software and made it available for others would be thrilled to see it used in a creative setting, and this might help attract more people to computational creativity. It takes real effort to build systems which rely on other people's software, but the benefits are much greater, as the power and flexibility of the software vastly increase.

Climbing the meta-mountain

Software is mostly a tool for humans to use, and until we can convince people that software can act autonomously for creative tasks, the general impression will remain that software is no more use to society than, say, a microwave. The main problem is that, even within computational creativity circles, we still build software that is intended to be used by, or at least guided by, us. A very common way in which this manifests itself is that we let the software have some autonomy in the production of artefacts (which may involve the software assessing artefacts), but we retain overall creative responsibility by choosing which artefacts to present to the world. Moreover, a criticism that people often level at so-called creative software is that it has no purpose. That is, if the human didn't run the software, analyse its output and publish the results, then nothing would happen - which is not a good sign that the software is creative. This is a very valid criticism. However, it is one that we can manage by repeatedly asking ourselves: what am I using the software for now? Once we identify why we are using the software, we can take a step up the meta-mountain and write code that allows the software to use itself for the same purpose. If we can repeatedly ask, answer and code for these questions, the software will eventually climb its meta-mountain, and will create autonomously for a purpose, with no human involvement.

The creativity tripod

In many domains, in particular the visual arts, how the artefact is produced is very much taken into account when people assess that artefact. This leads to a genuine, and understandable, bias towards human artefacts over computer generated ones, and this feeling is impervious to any silly Turing test (see below) which shows *scientifically* that people can't tell the difference between human and computer generated artefacts when they are presented out of context. This isn't a fatal problem, though, as long as we are happy to manage the public's impression of how our software works. Most people don't realise that modern software is a hugely complex and multi-faceted thing. Rather, they may harbour the impression that software is essentially a collection of largely simple algorithms. This leads to three main criticisms about computational processes, namely that: they lack skill; they lack appreciation; and they lack imagination. We should therefore manage these misconceptions by describing our software along these dimensions. Moreover, we should regularly ask ourselves: if I were to describe my software using this supporting tripod of creativity terms, where would the weakest link be? In identifying and addressing such weak links, we can build better software both in terms of what it is able to achieve practically, and what it appears to be doing. Managing the perception of creativity in software is as important as building more intelligent algorithms in domains where cultural, contextual and historical precedents play an important role. So, if you have software which doesn't appreciate its own work, or the work of others, or its subject material, etc., then write code which achieves this. If you have software which isn't particularly inventive - think hard about how some routines which could be described as imaginative can be implemented. Using this tripod, I've managed to devise a base-line test for creativity in software which I'm happy to defend. That is, if the software is producing artefacts with at least one behaviour that could genuinely be described as skillful, one that could be described as appreciative, and one that could be described as imaginative, then the software should be described as creative. There are two caveats here: firstly, this is not a prescription for creativity in people; secondly, this is a base-line test, i.e., it doesn't mean that the software is highly creative, indeed, it is our responsibility to keep adding skillful, appreciative and imaginative behaviours so that the software is perceived as increasingly creative.

Beauty is in the mind of the beholder

Turing-test style experiments may seem attractive because it shows some level of success if the artefacts being generated by your system are vaguely comparable to those produced by a person. However, computers are not humans. This should be celebrated. We should be loud and proud about the fact that our software is generating artefacts that humans might be physically able to produce, but might not choose to produce. More to the point, we should never hide the fact that the artefacts are generated by a computer, because this kind of deception can set the computer up for a fall. For instance, imagine a Turing-tester saying: “And so, I can now reveal that these are the paintings produced by a human artist, and these are the paintings produced by a convicted murderer”. While this example may be a little crass, I think it makes the point: by stating that the aim is to produce artefacts which look like they might have been created by a person, it explicitly lowers the value of the artefacts produced by computer. By using Turing-style tests, we are admitting that a pastiche is all that we aim for. At best, this shows that we don’t understand one of the fundamental purposes of creative endeavours, which is to produce something which no-one has produced before. In many domains (including some scientific ones), there is no right or wrong, there is only public opinion and the values of influential people in that domain. As there is no reason why we can’t change that opinion, there is no reason why we should compare our computer generated artefacts to those produced by people. We can change the mind of the beholder to more appreciate the value of the artefacts produced by our software, and in trying to do so, we can learn a lot about the general perception of creativity in society.

Good art makes you think

I’m not going to delve into the debate about what art is here. It’s difficult to argue against the fact, though, that scientific discoveries force us to think more about the Universe we inhabit, and many of the best works of art/music/literature were explicitly designed to force the viewer/listener/reader to engage their brains more than usual. Sometimes, the artworks are designed to make most people engage their brains in roughly the same way, other times the artworks are meant to be interpreted in many different ways. Sometimes, the purpose is to engage people on a cognitive level, other times the purpose is to engage them on an emotional level. We write software to produce artefacts like these. Hence, our software should produce artefacts with the explicit purpose of making the human audience think more. This can be achieved in a number of ways (disguise, commentary, narrative, abstraction, juxtaposition, etc.), and some of these are easier to achieve than others. More than any other aspect of computational creativity research, this sets us apart from researchers in other areas of AI. In these other areas, the point of the exercise is to write software to think for us. In our area, however, the point of the exercise is to write software to make people think more. This might help us argue against people who are worried about automation encroaching on intellectual life: in fact, in our version of an AI-enhanced future, our software forces us to think more rather than less.

In conclusion

It is tempting to hide from reality by convincing ourselves that as we write increasingly sophisticated software, the value of the artefacts it produces will increase in the eyes of society. In science, there is objectivity to a certain degree, but in the arts there is no collective consensus about good or bad art. Indeed, if any consensus came about, innovative artists would be expected to willfully subvert this notion of right or wrong. So, for most of us, our software produces artefacts which are ultimately assessed in a very subjective fashion. We could fragment the domain in order to derive more objective measures like they have in machine learning or theorem proving, but then - like them - we would probably never get around to putting the pieces back together again. Instead, we can begin to consider the public’s perception of the creativity - or lack of it - in computers. We can try and change these perceptions, and we can write and describe our software with them in mind, so that the bias against artificially created artefacts lessens and eventually disappears. I’m hoping that these catchy phrases will catch and become catchphrases. These - and others to come later - could provide a scaffolding around which we can build a united front which identifies us as a major area of AI research, enables us to engage meaningfully with the scientific and artistic worlds, and allows people to see the true beauty of the artefacts our software produces.