A unified theory and model of evaluation of creativity

Description of proposed research and its context (1 page)

Computational creativity (CC) is coming – if not here already. Evaluation in this area is fragmented and often ad hoc. In this project, we will build a software system that can itself contextually evaluate and justify its creative choices, autonomously or in dialogue with users.

Targeted advance Our focus is on software that can evaluate and interpret works created by others or by itself, and judge whether a given step is creative or meaningful. This would result, for example, in automatic programming tools that can write *good* software. One challenge is that current techniques of evaluation are hard even for people to apply. Evaluation strategies need a history of actions undertaken, a wide array of generated artefacts, or both, together with an explanatory theory. We will demonstrate a computational system that can build interpretive frameworks and assessment rubrics on the fly.

Why this matters With increasing automation, we need systems we can trust to make good decisions.

Budget £250,000 for exploratory research in computational creativity.

Keywords: interpretation, evaluation

***

How can we evaluate an *interpretation*? For example, sheet music gives you a kind of algorithm. Gifted musicians show us that a creative work can take on a life of its own in performance. This generalises. According to Umberto Eco, "a text is like a musical score," and fictional characters can also "become individuals living outside their original scores," or, more formally: "a fictional character is a semiotic object" [1]. People make chains of interpretations about fictional worlds and other creative objects. In the first instance, the validity of such interpretations is not "grounded" in real-world facts, but in shared imaginings, (fallible) perceptions, beliefs, assumptions, and other features of the interpreting agent. When instigating behaviour, certain interpretations may be deemed likely, based in part on a preliminary interpretation of the audience [2]. In computational creativity there has not been much work done on interpretation (outside of musical performance). And yet, interpretation is in many ways fundamental for contextually appropriate creative behaviour. Furthermore, systems that can draw on large datasets may not have the need to be particularly creative. In many cases this is quite useful: the more we know about an area, the more we can generate ever-more-accurate depictions of that area, like displaying a photograph at higher and higher resolutions. But what about situations where we need to think outside the box? For example, consider the thinking that lies behind "100 tweets summarised in 1 pithy comment." This task is likely to require creative interpretation. Generative software based on text mining may be able to add the $(n+1)^{\text{th}}$ incremental contribution. But if we require an "aha moment" then CC is the right field to turn to. We can, for example, consider a "third way" in natural language understanding that builds on ideas from text mining and retrieval, but that offers new kinds of understandings.

> "I noticed that Churchill makes use of rhyme in his speeches, so I made some rhyming couplets of positive and negative words in his style."

How would we evaluate this kind of understanding? For example: what is the poem like, what were the rules by which it was created, and above all, is it fit for purpose?

References

[1] U. Eco. On the ontology of fictional characters. In: *Sign Systems Studies* 37.1/2 (2009), pp. 82–97.

[2] P. Kockelman. The Ground, The Ground, The Ground: Or, Why Archeology Is So 'Hard'. In: *The Yearbook of Comparative Literature* 58 (2012), pp. 176–184.

Background (1 page)

Our work will be carried out in the context of automatic programming oriented towards the target domain of computer poetry. In particular we will solidify our recent advances with the FloWr system [1]. FloWr is a modular tool that client systems can use to connect nodes in a process flowchart in an *efficient* and *effective* manner. The FloWr system makes information available to show which inputs and outputs the nodes expect (so that meaningful connections can be formed). A key innovation in this project is that the system will be able to contextually evaluate individual steps on the way to a given goal.

The discussion of interpretation presented above, together with recent advances with FloWr and the availability of standard tools for language processing motivates a set of experiments based on concrete "acceptance criteria" for textual products and automatically generated software processes. One source of acceptance criteria would be basic exercises in poetry, or some other set of writing exercises. The process of responding to a writing prompt is something like this: (1) Read and understand the "prompt" to a sufficient degree; (2) compose a response that "makes sense"; (3) criticise the response along various dimensions, for instance, does it read well, does it tell a story or develop a character?; (4) how might it be improved? Note that since we are talking about *computer*-generated poetry, not only can we critique the outcome, we can also criticise the process, and read the product against the process (or vice versa). In addition to writing a response to a prompt, another task must be solved: the computer must decide *how* to respond to the prompt. We can compare this with Quillian's classic work on "The Teachable Language Comprehender: A Simulation Program and Theory of Language" [2]. Quillian's program took the novel – and fundamental – approach of understanding things in such a way that the new understandings could be added directly to its knowledge base.[1] When reading a piece of text, the TLC program would use so-called "form tests" to extract information from the text, and would then search its memory for related information that can be used to make sense of the input data. Now-standard approaches in poetry generation continue TLC's focus on understanding language semantics through grammar [3].

Reasoning about the manner of composition as well as the content of the response will add a new multimodal aspect to text understanding and knowledge representation. To be sure, a representation of poetic process in this form would be different from the embodied sensory-motor experience of humans. The program we envision would be able to transform a poem it into a program (or family of programs) that can generate an adapted version of the input poem in its own words, or another reply, and that can justify its steps. This should be seen as effectively the opposite of earlier work in computer poetry that sought to mimic a poet's surface style, without extending to *meaning* [4]. To this end our system will employ corpus methods rather than focusing exclusively on grammar.[A][B]

References

[1] J. Charnley, S. Colton, and M. T. Llano. The FloWr Framework: Automated Flowchart Construction, Optimisation and Alteration for Creative Systems. In: *Proceedings of the International Conference on Computational Creativity*. Association for Computational Creativity, 2014.

[2] M. R. Quillian. The Teachable Language Comprehender: A Simulation Program and Theory of Language. In: *Commun. ACM* 12.8 (1969), pp. 459–476.

[3] H. Gonçalo Oliveira and A. Cardoso. Poetry generation with PoeTryMe. In: *Computational Creativity Research: Towards Creative Machines*. Ed. by T. R. Besold, M. Schorlemmer, and A. Smaill. Atlantis Thinking Machines. Atlantis-Springer, 2015. Chap. 12, pp. 243–266.

[4] J. P. Carslisle. Comments on Kurzweil's Cybernetic Poet. In: *Americas Conference on Information Systems (AMCIS) 2000 Proceedings*. Ed. by M. Goul, P. Gray, and H. M. Chung. Association for Information Systems, 2000.

[1] In practice, "While the monitor can add TLC's encoded output to the program's memory, the program itself makes no attempt to do so, nor to solve the problems inherent in doing so" [2, p. 473].

[A] EdNote: **Joe@Stephen**: Please elaborate on vector space models.

[B] EdNote: **Joe@Simon**: Can we dispense with the bibliographic references here for more room?

National importance (1/2 page)

We have recently seen the impressive results of repeated practice in a restricted domain that has previously been dominated by humans, namely in demonstration games played by Google DeepMind's AlphaGo software. However, as Will Knight, the Senior Editor for AI in the MIT Technology Review remarks: "Some argue that a better way to gauge progress towards general AI is to ask computers to take much broader and more complex challenges." In the same publication, Doug Lenat: "It's fine to say we'll have programs that excel at checkers and chess and Go, but that's very different from saying those programs will be able to have prolonged conversations that cause you to make decisions involving human life."

The approach in this project focuses on support fo computer systems that learn, either independently from text, or through dialogue. We will draw on standard NLP techniques, but will combine them with a radically new approach to text understanding. There will be wide applications, for instance in education. Thus, this project could contribute to fulfilling the recent prognostications of Bill Gates: "I think we will get tools like personalized learning to all students in the next decade."

Meaning is not just a matter of text, however, and the approach we develop here will make fundamential contributions to automatic programming, by developing computer programs that can reason about code. Thus, a concrete outcome from this project would include key steps towards a "Stack Exchange for computers" – a system that would fulfil Turing's idea that machines should "be able to converse with each other to sharpen their wits."

Academic impact (1/2 page)

To reiterate the core themes of the proposal: (I) If agent A can respond to a writing prompt in an appropriate way, it demonstrates that A understands the "problem" that is presented by the writing prompt; (II) A's textual response should be a "solution" to that problem that is built from things like images, characters, plot points, and relationships between them; (III) In order to compose the textual response, A needs to solve an "automatic programming" problem, that is, A needs to figure out *how* to compose a solution; (IV) A should respond appropriately to critique, typically by adapting its programmatic interpretation.

In order to address the automated programming problem at the heart of this challenge (III), it is likely to be important to expand on the bespoke data-type and supplementary meta-information, e.g. minimum values, that FloWr currently supports. Following the methodology of "Design by Contract" [1], node authors would be able to make explicit statements of pre-conditions, post-conditions, and invariants. Sophisticated automatic programming clients could then reason about these specifications. There are a range of Java libraries that support this sort of annotation, e.g., the Java Modelling Language [2]. Using such an approach, we plan to see whether an automatic programming tool could rediscover hand-crafted patterns from existing FloWr flowcharts. This would be informed by related work on reasoning about formal specifications, much of which carried out in connection with theorem provers such as Coq [3, 4]. The text understanding problem, (I), will also require novel techniques. However, sticking with text solves a lot of problems and we will be able to draw on relevant prior art in text understanding, e.g. for essay-marking. Contributions to (II) and (IV) will advance the relationship between semiotics and computation.

References

[1] R. Mitchell and J. McKim. *Design by Contract, by Example*. Addison-Wesley, 2002.

[2] G. Leavens, A. L. Baker, and C. Ruby. JML: a Java Modeling Language. In: *In Formal Underpinnings of Java Workshop (at OOPSLA'98)*. 1998.

[3] A. Bove and V. Capretta. Computation by prophecy. In: *Typed Lambda Calculi and Applications*. Springer, 2007.

[4] P.-N. Tollitte, D. Delahaye, and C. Dubois. Producing certified functional code from inductive specifications. In: *Certified Programs and Proofs*. Springer, 2012, pp. 76–91.

Research hypothesis and objectives (1 page)

Hypothesis *Producing a poem-writing algorithm is a creative interpretation of a given text that can help us pinpoint where computational models of meaning need to be refined.*

In short, similar to "The Painting Fool sees!" [1], the outcome of this project will be "FloWr reads and evaluates!" The program will be different from standard machine learning approaches that seek to copy surface style but which do not have a real grasp of content. In each instance, the aim of adapting a given text is to bring into existence a new algorithm based on what someone (like Churchill for example) would put into his or her text. The computer will be able to evaluate and describe its choices in this regard, and training will progress at the level of meaning.

Progressive nature of the problem

It will be possible to address this the core issues progressively, dealing with greater and greater fidelity with things like *apprehension of textual features*, *style of response*, and *further adaptations to process in light of feedback*. Text adaptation (e.g., a story to a play) has been used in an English-language teaching context [2], and we will follow these outlines, using an agent-based model in which the agents learn new skills by practicing text understanding and generation together. This will be measured both through programmatic means, and through expert and popular evaluations of the work (e.g., at poetry readings and web dissemination). Our first milestone will tackle elementary poetic forms designed to build literacy; we will aim to conclude the project with a compilation of poems that are of interest to professional poets.

Overview of research trajectories

WP1 *Framing layer.* In the fellowship (EPSRC Grant EP/J004049/1, Computational Creativity Theory) Colton and Charnley focused on the generative side of FloWr. In this project, John would integrate 2 additional systems. A layer that can observe what's going on and comment on it, and another layer would facilitate evaluation on top of the accountability that comes through commentary. FloWr (and clients) should be accountable for generated poetry, flowcharts, and code. This needs server-side support in terms of a standard toolkit and reasoning workflow.

WP2 *Linguistic layer.* McGregor and colleagues and Queen Mary have been doing things with vector models and poetry. We will draw on this and work done within the WHIM project to… and will contribute to evaluation with experts.[C] FloWr needs to be extended with additional modules that will enable it to read and generate poetic texts, interfacing through an automatic programming layer.

WP3 *Testing layer.* In the context of the COINVENT project, Corneli has developed a FloWr API client in Clojure that makes use of FloWr's text-processing and automatic programming capabilities. In this project, that work will be extended of *software testing* (e.g., *assertions*, *advice*, and *contracts*) in order to provide ways for (evolving) programs to interact with their context. Methods for reflection and evaluation of creativity from psychology will be applied to develop a context-sensitive client that can create, run, and assess flowcharts in FloWr.

WP4 *Evaluation.* All of the researchers will contribute to evaluation at different at the level of theory, output, holistic system features, system development trajectory, and online system behaviour. A core goal for the project is to "embed" evaluation within the system; thus, our evaluation work will include the standard work with domain experts (poets), and additional programming work.

References

[1] S. Colton, J. Halskov, D. Ventura, I. Gouldstone, M. Cook, and B. Pérez-Ferrer. The Painting Fool Sees! New Projects with the Automated Painter. In: *Proceedings of the Sixth International Conference on Computational Creativity, ICCC 2015*. Ed. by S. Colton, H. Toivonen, M. Cook, and D. Ventura. Association for Computational Creativity, 2015.

[2] L. Raw. The Paragogy of Adaptation in an EFL Context. In: *Teaching Adaptations*. Springer, 2014, pp. 26–40.

[C] EdNote: **Joe@Stephen**: Please help flesh this out.

Programme and methodology (2 pages)

Dimensions of evaluation We will draw on these dimensions both in the evaluation of our work (see publication targets below), and to "embed" within the system (WP4T3): *output* (via audience or consumer opinions); *system development* (via assessing programmer and machine contributions to code); *unexpected (online) behaviour of the system* (via population-based measures of agent behaviour); *holistic features of the system* (via probes that examine qualitative features associated with perceptions of creativity); *contributions to theory* (via judgements of explanatory power and other attributes).

Development milestones, tasks, and publication targets This plan follows a well known five-stage model of creativity: *preparation*, *incubation*, *insight*, *evaluation*, followed by *elaboration* [1, pp. 79-80].

M1 Understand prompts to a sufficient degree to generate meaningful responses. *Criterion*: FloWr can tackle simple poetic exercises [2].

- WP1T1 FloWr needs a more robust library of Design by Contract (DbC) style interface for nodes, together with reasoning support for clients.
- WP2T1 We need nodes that can read various features of poetic texts and understand simple narrative instructions.
- WP3T1 FloWr clients should be able to reason about FloWr's DbC descriptions, and apply instructions to generate flowcharts.
- WP4T1 Framing our research questions and methods with reference to the relevant philosophical literature (e.g. on intentionality).
- WP4T2a Preliminary user study focusing on content.

Publication: For a journal on AI and language, month 7. Summarise initial lingustic capabilities and automatic programming framework; evaluation in terms of *system development* and preliminary *output*.

M2 Critical reflection on a family of responses to describe their properties. *Criterion*: FloWr can address creative writing prompts and critique the texts it generates.

- WP1T2 FloWr needs to support a notion of goal satisfaction for flowcharts.
- WP2T2 We need nodes that can generate text using background knowledge that has been extracted from repositories and stored in vector space (or other) models.
- WP3T2 FloWr clients should test generated texts against constraints, closing the loop between "reading" and "generation."
- WP4T3 After each technical milestone and paper is completed, work should be carried out in the following milestone to add more evaluation nodes summarising what we learned.[2]

Publication: For a journal on automated programming, month 13. Describe FloWr's improved automatic programming support; evaluation in terms of *system development* and *system behaviour*.

M3 Code or text generation to solve a problem posed by critical reflection. *Criterion*: A FloWr client can generate flowcharts that "adapt" poems in various styles.

- WP1T3 FloWr needs core support for loops, recursion, nested flowcharts, etc.
- WP2T3 We need to read and write textual markup for adaptation of poems.
- WP3T3 FloWr clients (agents) should adapt their behaviour based on contextual feedback.
- WP4T2b User study with experts, focusing on system behaviour.

Publication: For a journal on AI theory and methods, month 19. Describe our work on "embedded evaluation" and context awareness; evaluation in terms of the *holistic features of the system*.

M4 Experimental process to evaluate result. *Criterion*: A FloWr client can choose and apply appropriate methods to critique texts and flowcharts.

- WP2T4 We need to be able to read free-form feedback and translate it into the above markup.

[2] Thus, this task repeats.

WP3T4 Critique generated and adapt flowcharts using methods parallel to those piloted for poetry.

Publication: For a journal on AI and language, Month 25. Explain similarities and differences between reasoning about code and reasoning about language; evaluation in terms of *system behaviour*.

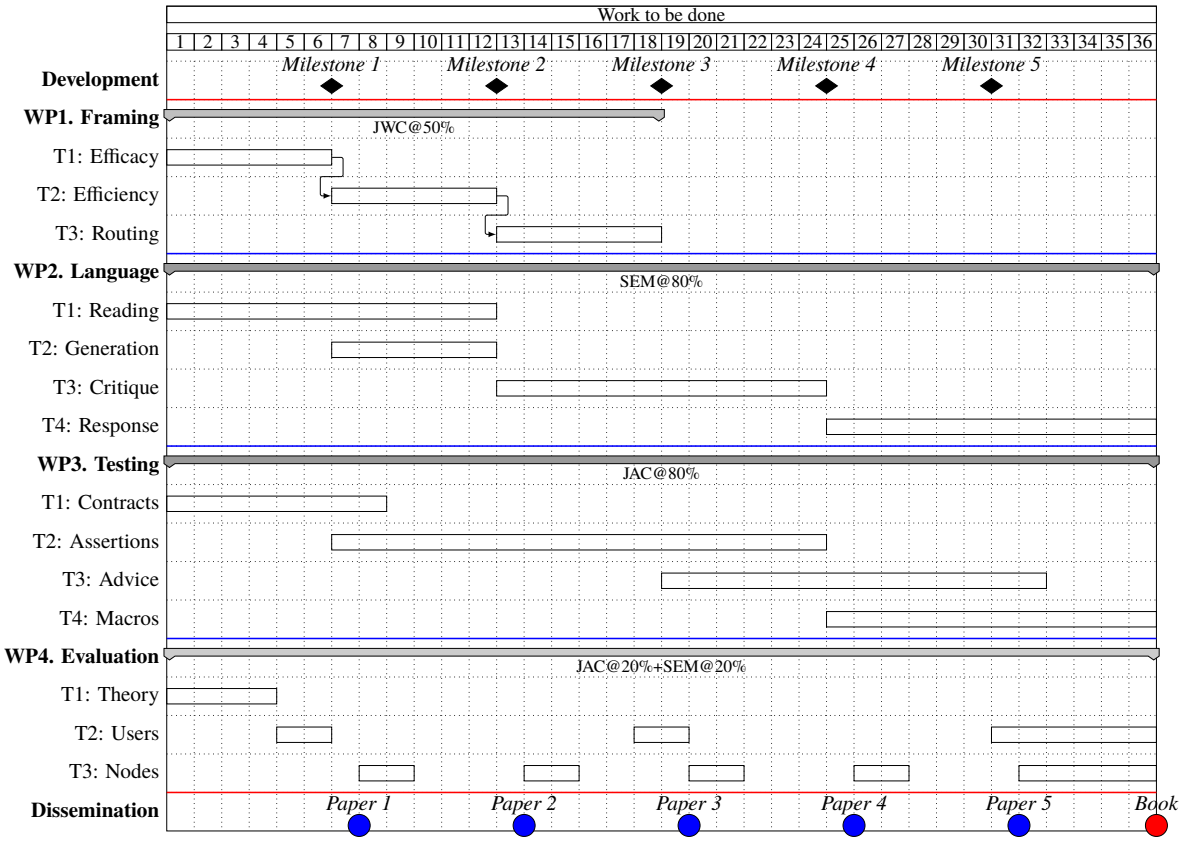
M5 Change the way the poetry generation process works in light of performance or other feedback measures. *Criterion*: A FloWr client can respond to critical feedback from an expert user.

WP4T2c A more intensive summative study will be deployed to evaluate system output. Findings will be recorded in the form of modular additions to the codebase that serve to facilitate a dialogue between users and system.[3]

Publication: Month 31. Last journal article, summarising and contextualising the project as a whole, framing future work; evaluation primarily in terms of *system development* and *theory* (new hypotheses).

Final publication: Month 36. We will publish a collection of machine generated poems with peer reviews and/or call-and-response poems written by professional poets; evaluation of final *output*. Examples of automatically generated programmatic material will be included in a technical appendix.

Gantt chart



Risk management Core infrastructure is front-loaded. We will use a modular, client/server approach to maintain a division of concerns. User studies and literate documentation are built into the project plan.

References

[1] M. Csíkszentmihályi. *Creativity: Flow and the Psychology of Discovery and Invention*. Vol. 39. Harper Perennial, 1997.

[2] V. L. Holmes and M. R. Moulton. *Writing simple poems: Pattern poetry for language acquisition*. Ernst Klett Sprachen, 2001.

[3] WP2T4 and WP3T4 continue during this phase.