Meta-Evaluating Quantitative Internal Evaluation: a Practical Approach for Developers

Filippo Carnovalini [1], **Nicholas Harley**[2], **Steven T. Homer**[2], **Antonio Rodà**[1], and **Geraint A. Wiggins**[2,3]
[1]University of Padova, via Gradenigo 6, Padova, Italy
[2]Vrije Universiteit Brussel, Pleinlaan 9, 1050 Brussel, Belgium
[3]Queen Mary University of London, Mile End Road, London E1 4NS, UK
filippo.carnovalini@dei.unipd.it

Abstract

Within the field of Computational Creativity, evaluation is one of the most important and more difficult tasks. Sometimes evaluation is part of the creative systems themselves, becoming an internal evaluation. Being a module of a creative system, it is useful to evaluate how effective this internal evaluation is.

In this paper, we propose a procedure for the (meta-) evaluation of internal evaluation modules, that allows for incremental development of both the evaluation module and the creative system, which are considered fully independent of each other. The procedure works by statistically comparing the evaluation of the average output of the generation system with the best results from the same, to see if the evaluation procedure can statistically distinguish the two. We then show how to apply the procedure giving one example evaluating a module we designed to assess structural coherence in generated folk music.

Introduction

Evaluation is one of the most important aspects of Computational Creativity, but this widely used term describes more than one issue that CC practitioners need to face. The most obvious evaluation problem that anyone who ventures into the field soon encounters, is that of evaluating the results of a creative system, and whether the system can be defined creative in its own (Jordanous, 2012). The difficulty of this task called for more precise definitions of what it means to be creative, and, more importantly, what it takes to be creative. This led to the realization that another evaluation problem arises: according to Margaret Boden and other prominent researchers, for a system to be called creative, it needs to be able to explore a conceptual space on its own, and to be able to evaluate what it encounters in non-previously explored areas of said space to find the better results and to guide the exploration (Boden, 2004; Ventura, 2017). This is the problem of what we will call Internal Evaluation – sometimes referred to as self-assessment (Lamb, Brown, and Clarke, 2018), reflection (Pérez y Pérez and Sharples, 2004), or appreciation (Colton, 2008): while generating *something* is trivial, being able to tell apart good and bad results is what can make a creative system of interest to the community (Agres, Forth, and Wiggins, 2016). This paper addresses both of these problems, by analyzing how an internal evaluation method can be assessed in itself (in what effectively becomes a meta-evaluation: Jordanous, 2014), to tell if the internal evaluation does indeed help in generating better results or if it only becomes another layer of unnecessary complexity.

The approach we propose has the advantage of being relatively simple to implement while developing the internal evaluation module, and is general enough to apply to any kind of generated artefact. Using an existing generation system, the developers must select some results that are considered better than average, and some results that are not cherry-picked at all and can be considered average results from the system. The internal evaluation is not applied to any of these results, meaning that the development of the evaluation module is independent and incremental to the one of the existing generation system, allowing for more flexible development. The artefacts that are thus selected constitute a dataset for the development of the evaluation method. The evaluation can then be applied to the artefacts in the two groups, and a statistical analysis is run on the results. If the two groups show a significant difference in results, the evaluation module can be considered satisfactory and can be implemented within the generation system. Otherwise, further development can be applied to the evaluation method without the need to change the system.

We exemplify this approach by applying a method for the evaluation of musical structural coherence to an algorithm for melody generation developed by other researchers: in doing that we do not evaluate the generation system in itself but only our own module for internal evaluation, showing how the two can be completely separate for this meta-evaluation procedure.

Procedure and Examples

In this section we will explain our procedure for meta-evaluation, giving a practical example of how we applied it to a realistic case-scenario, and also giving some indications on how this can be implemented within an incremental development cycle.

Requirements

The proposed procedure, while being general enough to adapt to a wide variety of algorithms and approaches, has

Proceedings of the 12th International Conference on Computational Creativity (ICCC '21)
ISBN: 978-989-54160-3-5

some requirements that need to be addressed before explaining how the procedure works.

The very first requirement is that the system in which the evaluation module needs to be implemented must already be functional. This means that the system must be capable of generating the desired artefact, although the quality of these artefacts might still be in general unsatisfactory (and thus calling for internal evaluation). This also means that the system should not depend in any way on the evaluation module, and the vice versa should be preferred. While this requirement might seem stringent to some, this will also allow for a modular implementation of evaluation modules as well as a facilitated incremental development. In some cases, one might be only interested in the internal evaluation rather than the generation of artefacts. In this cases, it can be appropriate to use a system developed by others that generates similar artefacts as those that are under exam. For this paper, that is what we did: instead of implementing a novel music generation algorithm, we used FolkRNN (Sturm et al., 2019), a readily available system for the generation of folk music. This shows the complete independence of the evaluation from the inner workings of the system, that we used as a black box in this work.

The second requirement is the evaluation module itself. Here we will not focus on the development of the evaluation module, which can be viewed in this procedure as another black box. The only condition is that it must accept the kind of artefacts that need to be evaluated as input, and output a quantitative evaluation, i.e., a number. Sometimes the evaluation needs to consider aspects that are "deeper" than the final artefact. In this case, either the evaluation module should be able to analyze them itself to keep the full detachment from the system, or the system must be modified to output those specialized information as well as the final output.

Procedure

Step Zero: Feature Definition. The developer should be completely clear about what feature of the artefact the evaluation module should evaluate: saying that it should find the "better" results might be too vague and lead to more biased and less informative results. While this step is more of a premise than an actual step, its importance justifies the inclusion in this section: this step drives the selection of samples for the dataset, as explained below, and not being clear about what is being evaluated will hinder the entire process (Pearce, Meredith, and Wiggins, 2002; Jordanous, 2012). It is also important to choose at this step if the meta-evaluation will be an Holistic Evaluation or a Specialized Evaluation, as explained below.

Step One: Dataset Creation. For this procedure to function, it is necessary to create an ad-hoc dataset of *output artefacts*. This should not be confused with the dataset that was used to train the generation system nor the one that was used to train the evaluation module (if there are such datasets): these are usually human-generated artefacts that are used to define a style or an objective for the system. In this case, the dataset should only be comprised of artefacts that were generated by the system itself. These must be divided into two groups: the "average" results and the "cherry-picked" ones. For the first group, a set of any N valid results created by the system selected at random (i.e., without any form of selection: ideally, the first N successive valid products of the system could be used) should be used, with N big enough to allow statistical analyses to be performed.

The second group must instead be only comprised of results generated by the system that are *deemed good*. This is an intentionally imprecise term, because it can depend on the objective of the evaluation, and we can distinguish two main types of evaluation, in this sense. An **Holistic Evaluation** will see how the evaluated feature impacts the results in term of general metrics such as creativity, value, style, novelty. A **Specialized Evaluation** will instead only evaluate the same feature as the one considered by the internal evaluation module as defined in step zero. These two kinds of evaluation are complementary: the first one is most useful when evaluating the effect of the internal evaluation module on the system in general, while the second is most useful when the internal evaluation module is being evaluated on its own. In both cases, the pieces that will form the second group must be selected by humans with knowledge about the generated artefacts, possibly the author of the system or a group of experts or via controlled questionnaires or crowdsourcing. The method for the selection can depend on the kind of evaluation and the evaluated feature: for example, if an Holistic Evaluation is being performed to find the effect on creativity, the selection could be based on the Consensual Assessment Technique (Amabile, 1983). Vice versa, if a Specialized Evaluation is being performed the author of the system might want to analyze the results on its own to check if they fit his idea of what the evaluation system is supposed to select. Regardless of what method is used, the system authors should be very clear on how the selection was performed both while designing the evaluation and while reporting its results.

Depending on the method used, creating this second group as large as the first group can be extremely time-consuming. Even if it is not possible to include N elements in the second group, the selection process should start from N elements, from which the selected results can be extracted. In this way, the numeric disparity between the two groups will reflect the distribution of good results in the average results.

Step Two: Evaluation This central step is rather self-explanatory. Each of the pieces selected in the the dataset must be evaluated through the evaluation module, and the results must be collected. As already explained, this procedure requires a full separation of the evaluation from the system, as is clear from the fact that the evaluation happens (long) after the generation is complete. This also allows for some final modifications of the evaluation module at this point, if the process points out some flaws in the software. Yet, the developer should restrain the urge to modify the code to fine-tune it to get better results. While this is feasible and sometimes useful, if the developer chooses to do so step one should be repeated and a new dataset created, otherwise the final evaluation results would be biased.

Step Three: Statistical Analysis The final step represents the actual assessment of the evaluation module. The results collected on the evaluation dataset represent two different populations, and are expected to be the result of two different probability distributions: if they originated from the same distribution, then it would be the case that there is no difference between the two groups seen from the eye of the evaluation module. Otherwise, if the evaluation module sees a difference between the two groups, there should be a statistically significant difference between the two. To evaluate this, we propose to use the Shapiro-Wilk test to check if the two distributions are *not* normal. In that case, a non parametric test like a Mann-Whitney U-test can be used to tell if the two distributions differ. If the Shapiro-Wilk test does not strongly indicate that the distributions are not normal, and there is enough evidence suggesting that the two distributions are indeed normal, the more powerful two-sample unpaired t-test (Welch's t-test) is to be preferred (Dodge, 2008). These suggestion outlines some of the more common two-sample tests, but other tests may be appropriate depending on the data. All these tests indicate, through the p-value, if the two distributions are statistically different. In general, a p-value lower than 0.05 can be considered a good result, but since we are dealing with artistic features, depending on the applications the developer can be satisfied with a p-value higher than 0.05, if for example there are other features to be considered in the final system or if the developer wants to leave room for "happy accidents" and allow for pieces that do not comply with the stricter definitions of the evaluation metrics, but could be considered good nonetheless and possibly more creative because of how they escape some rules. For this reason, the researchers might aim for less restrictive values (such as $p < 0.1$) or use the p-value as a continuous metric rather than a boolean one to compare the results of different iterations of a system or different systems. Once again, regardless of the chosen objective, it is important to establish clear goals for this value beforehand to establish if the evaluation can be considered satisfactory or not.

Example

In this section, we review the above procedure by applying it to a specific example to further discuss problems that can arise and other caveats. In doing this, we will meta-evaluate an evaluation method for music generation.

Step Zero: Feature Definition. Our evaluation module tries to evaluate how structurally coherent a piece of generated music is. While it is not important to know the inner workings of the evaluation to apply this procedure, we will briefly discuss how the algorithm works for completeness. This module accepts as input a monophonic music piece with chord annotations, and segments it into fragments that last two measures. The algorithm then recursively simplifies each segment in a manner inspired by Schenkerian Analysis (Simonetta et al., 2018), and then operates pairwise comparisons between the trees constructed from each segment. From all the comparisons operated on a corpus of folk reels from the Nottingham Dataset (Foxley, 2011), some probability distributions that show how those comparisons typically develop are constructed. The new pieces that need to be evaluated are similarly analyzed, constructing trees to describe the new pieces. The evaluation of the piece uses these trees, but instead of constructing probability distributions, we compute the Information Content of the piece's trees when compared to the learned distributions' trees. The Mean Information Content, the metric we use in this example, thus serves as an indication of how unexpected (and thus non-typical) the structure of the new piece is when compared to the learned corpus since it compares how common these trees (that represent structure) are with respect to the analyzed corpus. More information about the specifics of this system is available in previous publications (Carnovalini et al., 2021b,a). It is worth noting that the choice of the dataset for the learning of the evaluation module is also to be considered within this point: in this example, it is reasonable to evaluate the output of FolkRNN with the typical structures of traditional Reels, but in other cases having the constraints imposed by a certain corpus could be too limiting.

For this example, since we developed the evluation module but we used FolkRNN, a system developed by others, for generation, we chose to perform a Holistic Evaluation, trying to assess if our evaluation module that evaluates structural coherence is able to impact the general value of the generated output as assessed by FolkRNN's own community (see next paragraph).

Step One: Dataset Creation. In our case the dataset was created via the FolkRNN web application. For the "average" group, we created ten melodies in the key of C major and 4/4 meter. For the other group, in order to simulate the choice of the system's creator, we decided not to select the songs ourselves but instead to select those pieces in binary time from the "Tune of the Month" section of the FolkRNN website, that contains pieces deemed most interesting by the developers and users' community. If we wanted to do a Specialized Evaluation on instead, we should have generated 10 songs and manually select (possibly with the help of expert musicians) the ones that show a good structural coherence.

Step Two: Evaluation Our evaluation system requires the harmony to be annotated on the generated melody, which is not present on FolkRNN generated pieces, so it was manually annotated. This is not ideal, but since we decided to use FolkRNN as a black box we could not modify it to automatically add harmonic annotations as we would have done on a system of our own. Moreover, our system evaluated structural coherence at a fixed length of eight bars, which is the typical length of a period in folk music. FolkRNN outputs pieces that are generally sixteen measures long (not considering repetitions) so each piece gave two samples for the evaluation.

Step Three: Statistical Analysis The "freshly generated" group includes 20 samples, while the "Tune of the Month" has 6 samples, due to the scarcity of songs in the homonymous section of the FolkRNN website. The two distributions of Information Contents seem non normal, according to the Shapiro-Wilk test and the inspection of the Q-Q plot, therefore we used the Mann-Whitney U-test to tell if the two

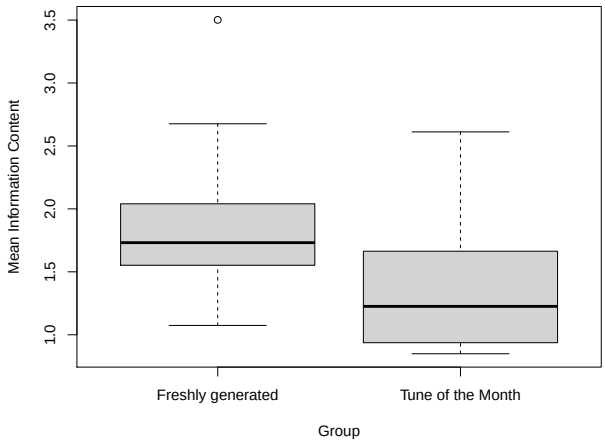

Figure 1: The boxplot comparing the Mean Information Content, i.e., the result of our evaluation module, of the samples that were generated specifically for this evaluation as average results and those that were selected by the community as 'Tune of the Month' in the past.

distributions are statistically different. The p-value resulting from the test is 0.053, a rounding error away from the 95% confidence interval that we aimed for. This is reflected in the boxplot of the two distributions (see Figure 1), that shows how the distributions have different means but also have a rather large overlap. In doing this evaluation, we found out that FolkRNN does a good job at giving a good structure to the pieces it generates, so it is not surprising that our metric is having a relatively hard time distinguishing the selected pieces from the non-selected ones. Figure 1 also highlights one outlier in the non-selected group. It is important at this point to also inspect such samples, as they could give more insights on possible hidden mechanisms and flaws of the evaluation method. In this case, the outlier has a repetition in the middle of the period, repeating the beginning of the two sub-phrases, which would be considered good structure, but fails to give good phrase endings. This tells us that our evaluation module sometimes gives more importance to phrase endings rather than phrase beginnings, an aspect that can be considered in future development.

Comments and Conclusions

In this contribution, we introduced a procedure for the meta-evaluation of quantitative evaluation methods meant to be implemented as internal evaluation modules within a computationally creative system. This procedure is based on statistical methods, and while it does require the developers to create an ad-hoc dataset for the evaluation every time a new method must be assessed, it does not necessarily require intervention of external experts, and allows for incremental development of both the creative system and the evaluation method. For all these reasons, we believe it can be useful and practical for researchers and developers at many stages of development. We showed how the procedure functions by applying it to a real case scenario, evaluating a module for the evaluation of structural coherence in folk music, applied to the results of FolkRNN. It is important to mention that this approach did not evaluate FolkRNN directly, nor we mean to make any sort of statement about that system. We only used it as a framework to evaluate our own evaluation module.

We did not discuss the development of the evaluation module nor of the system, since our procedure is meant to apply to a variety of situations, but the question on how to use the results of such an evaluation module within a system might arise. While the answer depends on the specific system, a few possible ways can apply to almost any situation. The most naive one is to use a threshold on the results of the module to discard any generated results that is too far from the ideal results, or similarly to use the results of the evaluation method to rank the output of the system so that the developer will first inspect those results that are most promising. When more than one internal evaluation feature is implemented, the developers might want to assign weights to every feature to create the final ranking, or might want to use regression algorithms on the evaluated features to find the overall best ranking. Having a variety of evaluated features might also give the possibility to specify settings for the generation/evaluation: in some cases the user might want to be stricter on some rules rather than others.

Regardless on how the evaluation modules are embedded in the final system, it is important to remind that the current proposal is useful for the meta-evaluation of some quantitative metrics used within the system to guide the generation process. On the contrary, this method is *not* a procedure for the evaluation of the system in itself and for the value or novelty of the final results. To that goal, human assessment done by experts of the artefact's field is still the method to be preferred (Jordanous, 2012).

Acknowledgments

FC is funded by a doctoral grant by University of Padova, and visited Vrije Universiteit Brussel thanks to a mobility grant from Fondazione Ing. Aldo Gini, which made this work possible. NH, ST and GW received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

We thank the anonymous reviewers whose comments allowed us to greatly improve this article.

References

Agres, K.; Forth, J.; and Wiggins, G. A. 2016. Evaluation of musical creativity and musical metacreation systems. *Computers in Entertainment (CIE)* 14(3):33.

Amabile, T. M. 1983. A Consensual Technique for Creativity Assessment. In Amabile, T. M., ed., *The Social Psychology of Creativity*, Springer Series in Social Psychology. New York, NY: Springer New York. 37–63.

Boden, M. A. 2004. *The creative mind: Myths and mechanisms*. London, United Kingdom: Routledge.

Carnovalini, F.; Harley, N.; Homer, S. T.; Rodà, A.; and Wiggins, G. A. 2021a. Studying structural regularities through abstraction trees. In *Manuscript Submitted for Publication*, 10.

Carnovalini, F.; Rodà, A.; Harley, N.; Homer, S. T.; and Wiggins, G. A. 2021b. A New Corpus for Computational Music Research andA Novel Method for Musical Structure Analysis. In *Audio Mostly 2021 (AM '21)*, 4.

Colton, S. 2008. Creativity Versus the Perception of Creativity in Computational Systems. In *AAAI spring symposium: creative intelligent systems*, volume 8, 7.

Dodge, Y. 2008. *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York.

Foxley, E. 2011. Nottingham Database.

Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4(3):246–279.

Jordanous, A. 2014. Stepping back to progress forwards: Setting standards for meta-evaluation of computational creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, 8. Ljubljana, Slovenia: Jožef Stefan Institute.

Lamb, C.; Brown, D. G.; and Clarke, C. L. A. 2018. Evaluating Computational Creativity: An Interdisciplinary Tutorial. *ACM Computing Surveys* 51(2):1–34.

Pearce, M.; Meredith, D.; and Wiggins, G. 2002. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae* 6(2):119–147.

Pérez y Pérez, R., and Sharples, M. 2004. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems* 17(1):15–29.

Simonetta, F.; Carnovalini, F.; Orio, N.; and Rodà, A. 2018. Symbolic Music Similarity through a Graph-Based Representation. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion - AM'18*, 1–7. Wrexham, United Kingdom: ACM Press.

Sturm, B. L.; Ben-Tal, O.; Monaghan, \.; Collins, N.; Herremans, D.; Chew, E.; Hadjeres, G.; Deruty, E.; and Pachet, F. 2019. Machine learning research that matters for music creation: A case study. *Journal of New Music Research* 48(1):36–55.

Ventura, D. 2017. How to Build a CC System. In *International Conference on Computational Creativity*, 253–260.