El arte detrás de los algoritmos

Ximena Molina Portilla
Susana Feng Liu

Por definición, ¿podría el software ser arte? El concepto de arte suele variar dependiendo de la fuente de la cual se obtenga su definición. Para efectos de este artículo, tomaremos como referencia la propuesta por Tolstói (2024) quien dice que “el arte no es una alegría, ni un placer, ni una diversión; el arte es una gran cosa. Se trata de un órgano vital de la humanidad que transporta al dominio del sentimiento las concepciones de la razón” (p. 140). Lo que distingue al arte, bajo esta perspectiva, no es la motivación interna de quien crea, sino la capacidad de la obra para transmitir emociones o ideas del mundo a quienes la experimentan. En este sentido, el software podría considerarse arte porque logra comunicar valores, emociones o intenciones a través de su diseño, funcionamiento e impacto en sus usuarios, no simplemente porque el programador lo haya creado con sentimiento o buena intención

El software debe tener aspectos con los cuales evaluar su calidad, de la misma manera que se evaluaría una obra musical o literaria. No se podrían utilizar los modelos de calidad existentes que ayudan a realizar esta evaluación porque, desde un punto de vista artístico, estos no tienen sentido. Además, hay que tomar en cuenta el tipo de software que se esté evaluando. Por ejemplo, un lenguaje de programación puede ser evaluado por su propósito, su originalidad, su nivel de ambigüedad, entre otros criterios. Con esto podemos confirmar que existen criterios aplicables al software, solo hay que moldearlos o crearlos de acuerdo al tipo del mismo.

Tomando el ejemplo anterior de los lenguajes de programación, podemos observar cómo este caso se asemeja a la literatura, ya que muchos de estos criterios son igualmente aplicables en ella. Al igual que en la literatura, el código es algo que se lee, puede confundir al lector, frustrarlo y generarle muchas otras emociones. Incluso más allá de su lectura, cierto software puede transmitir valores culturales concretos: un sistema desarrollado en una lengua indígena, por ejemplo, no solo cumple una función técnica, sino que visibiliza comunidades históricamente marginadas y evidencia

tensiones relacionadas con la inclusión tecnológica. Esto nos inclinaría aún más hacia la idea de considerar el software como arte, pero cabe destacar un área específica que últimamente ha dado mucho de qué hablar y donde quizás empecemos a dudar de nuestra postura: la inteligencia artificial.

¿Podría un producto desarrollado por IA ser considerado arte, aunque esta no posea sentimientos y, sobre todo, un alma?

En los últimos años, se ha generado un gran debate con respecto a la inteligencia artificial, específicamente la generativa. Si entendemos el arte como aquello que logra transmitir emociones e ideas, habría que preguntarse si un producto generado por IA es capaz de provocar ese efecto en quien lo recibe. Sin embargo, hay que recordar que la IA no se genera sola: debe haber un ser humano que construya un prompt que dispare la generación de una imagen, video, texto o audio. En este sentido, se ha pensado que la IA constituye una capa adicional de abstracción dentro del proceso creativo, de manera análoga a otras herramientas como compiladores o frameworks. La intención y los valores del desarrollador no desaparecen, sino que se expresan a través del uso de estas tecnologías, por tanto, lo que se comunica a través del resultado podría, en principio, seguir siendo de origen humano.

No obstante, hay un punto importante que, a pesar de no estar incluido en la definición de arte, hace la diferencia: la originalidad. En mi opinión, este aspecto es el punto de quiebre de la discusión. Hay que recordar que la manera en que funciona la inteligencia artificial “es esencialmente estadístico: aprende patrones, distribuciones y relaciones de palabras, imágenes o sonidos para generar resultados coherentes a nuevas entradas” (Aguilar, 2025). Todo lo que genera la inteligencia artificial es un conjunto de elementos que “aprendió” que, en otras palabras, significa que extrajo de algún elemento de una base de datos. De acuerdo con Osorio Umaña (2025), un objeto solo puede considerarse original si refleja la personalidad de su autor a través de elecciones libres y creativas. En este sentido, cuando el proceso de creación está condicionado por exigencias técnicas, reglas u otras limitaciones que anulan dicha libertad, la jurisprudencia del Tribunal de Justicia de la Unión Europea concluye que el

objeto carece de la originalidad necesaria para ser protegido como obra (p. 6). Bajo este criterio, la inteligencia artificial no puede ser considerada original.

Ahora bien, vale la pena cuestionarse si otros tipos de software enfrentan el mismo problema. La respuesta es no. Por ejemplo, los videojuegos, especialmente aquellos desarrollados por un equipo pequeño de personas. Existen muchos juegos cuyo propósito es concientizar sobre algún tema específico, y esto se suele lograr desarrollando mecánicas distintivas que los hagan únicos y memorables. Crear un juego con una idea original es totalmente necesario en este ámbito. A diferencia de la IA, estos softwares no toman decisiones creativas por sí mismos, sino que ejecutan directamente las instrucciones del programador.

¿De quién es el arte cuando el código es de todos?

El tema de la originalidad trae a la mesa un aspecto importante del arte: este suele tener dueño. Un pintor firma su pintura una vez que la termina para mostrar que fue él quien la creó, una canción siempre trae una sección de créditos para rendir homenaje a las personas que la crearon, pero ¿y el software? Es cierto que se puede mostrar al público ejecutables con sus respectivos créditos, sin embargo, ¿qué pasa con el software de código libre? ¿Acaso este representa una amenaza al arte?

El hecho de que un código sea libre, significa que, cualquier persona pueda modificarlo o utilizarlo, no implica que carezca de autoría. En el software de código abierto existen diversas formas de reconocer la contribución de quienes participan en su desarrollo. Una de las más comunes es a través del historial de *commits* en plataformas como GitHub o GitLab, donde cada cambio queda registrado con el nombre del autor. Además, según la guía en línea de *Open Source Guide*, se recomienda que en los proyectos se incluyan archivos de documentación como *CONTRIBUTORS.md* o secciones de agradecimientos en el *README.md* (GitHub, s.f.), que funcionan como espacios formales para dar crédito a los programadores y colaboradores. De esta manera, aunque el código sea compartido libremente, la autoría y las aportaciones individuales permanecen visibles y reconocidas.

Ahora bien, bajo el entendimiento de que el código puede llegar a ser arte cuando existe intención, propósito, sentimiento y una estructura cuidadosamente construida, en ese caso no podríamos necesariamente clasificar al código abierto como una amenaza para el arte en sí mismo. Por el contrario, podríamos decir que esta forma de desarrollar software llega a modificar la forma en la que entendemos la autoría y la creación artística.

Por ejemplo, si se compara con la música, un músico puede crear una pista considerada completamente propia a partir de demos, samples o melodías ya existentes. Aunque parte de elementos previos, la obra final sigue siendo suya porque la transforma, la reorganiza y le imprime su propio estilo. Del mismo modo, el código abierto parte de una base existente, pero cada desarrollador aporta algo de sí mismo: una solución, una mejora, una forma distinta de organizar el sistema o incluso una visión diferente del proyecto. Por supuesto, siempre y cuando se realice bajo los lineamientos con los cuales fue creado el proyecto de software libre. Puesto que no necesariamente todos admiten contribuidores (Open Source Guide, s.f.), que en ese caso la modificación del código fuente ya no se consideraría una contribución y podría llegar incluso a considerarse una copia más.

Bajo esta premisa, el código abierto puede entenderse como una forma de arte de tipo colaborativa. A diferencia de otras disciplinas donde la participación de varias personas puede diluir la autoría, en el software libre esta se conserva y se documenta. Como habíamos mencionado anteriormente, cada contribuidor recibe reconocimiento mediante *commits*, historiales de versiones o documentos dentro de plataformas como GitLab. De esta manera, el proyecto sigue teniendo autores, pero estos no son una sola persona, sino una comunidad de individuos que enriquecen constantemente la obra original. De manera que la nueva versión es una nueva obra original.

Además, esta colaboración suele surgir de una motivación distinta a la puramente económica. Tal como describe González Barahona et al. (2008), “el modelo de desarrollo en proyectos de software libre suele ser más informal, debido a que gran parte del equipo de desarrollo realiza esas tareas de manera voluntaria y sin recompensa económica” (pp. 111-112). Es decir, los contribuidores no participan porque

esperan una ganancia inmediata o porque desean convertir el proyecto en un producto para vender, sino porque buscan mejorar una herramienta que consideran útil y ponerla al alcance de más personas. Existe una intención genuina de aportar, perfeccionar el código y permitir que cualquiera pueda utilizarlo, adaptarlo o incluso continuarlo.

Esta idea la podemos relacionar con ciertas formas de arte que adquieren valor precisamente cuando dejan de estar limitadas a unos pocos. El arte callejero, por ejemplo, existe en espacios públicos y puede ser visto por cualquier persona, sin importar su condición económica o social. Básicamente se toma el “espacio público como un lugar donde la interacción se basa en la posibilidad de expresar las diferencias, la pluralidad de discursos entendiendo lo "público" como un espacio común a todos, compartido por todos.” (Levit y Pérez Roig, 2024, p. 75). Entonces, de la misma forma en la que el arte callejero se legitima en un espacio público accesible por todos, el software libre bajo esta perspectiva también obtiene valor cuando puede llegar a la mayor cantidad de personas.

Por otro lado, bajo este contexto, algunas personas podrían argumentar que abrir el código le resta valor, ya que cualquiera puede acceder a él, modificarlo o reutilizarlo. Sin embargo, esa crítica parte de una concepción del valor basada en la exclusividad. El principio mismo del código abierto consiste en que el conocimiento sea accesible para todos y en que la tecnología pueda evolucionar gracias a la participación colectiva. Por ello, también es válido pensar que el código abierto adquiere un valor mayor conforme llega a más personas, del mismo modo en que una obra artística puede volverse más significativa cuando logra inspirar, ayudar o transformar a una comunidad entera.

Referencias

Aguilar, O. (1 de julio, 2025). *Inteligencia artificial generativa vs. el potencial del cerebro humano: ¿Qué nos hace únicos?* Sanfer. https://sanfer.com.mx/blog/inteligencia-artificial-generativa-vs-el-potencial-del-cerebro-humano-que-nos-hace-unicos

GitHub. (s.f.). *Open Source Guide*. https://opensource.guide/

González Barahona, J., Seoane Pascual, J., y Robles, G. (2008). Introducción al software libre, febrero 2008. Universitat Oberta de Catalunya. https://openaccess.uoc.edu/server/api/core/bitstreams/f0f98d43-f6cf-45e6-abb6-1b58e68825c3/content

Levit, E. L., y Pérez Roig, P. (2024). Arte callejero: una herramienta de transformación social y del entorno. En M. Jalo (Coord.), Reconstruir la memoria pedagógica: experiencias docentes del departamento de Lenguas Modernas del Colegio Nacional (pp. 75-84). Editorial de la Universidad Nacional de La Plata (EDULP). http://sedici.unlp.edu.ar/handle/10915/170095

Osorio Umaña, F. (2025). El concepto de originalidad del Tribunal de Justicia de la Unión Europea y la inteligencia artificial. Revista Justicia & Derecho, 8(1), 1–11. https://doi.org/10.32457/rjyd.v8it.2873

Tolstoi, L. (2024). *¿Qué es el arte?* Ediciones Clío. https://doi.org/10.5281/zenodo.13958192

https://revistas.uautonoma.cl/index.php/rjyd/article/download/2873/2102