2023

¿LA IA VA A REVOLUCIONAR EL DESARROLLO DE SOFTWARE?

Matthew Belcher
y Danish Javed

codurance

Transformación del sector

En el mundo de la tecnología, en constante evolución, estamos entrando en una nueva era con la llegada de la Inteligencia Artificial Generativa (o Gen AI). La Gen AI promete redefinir la forma en que abordamos el desarrollo de software, automatizando las tareas rutinarias de codificación, generando estructuras de código complejas y mejorando la calidad del software. En este contexto, ¿está la Gen AI preparada para revolucionar el desarrollo de software? ¿Son sólo exageraciones o estamos realmente en la cúspide de algo transformador?

Las herramientas de Gen AI podrían aumentar drásticamente la productividad, reducir los errores manuales y desencadenar una oleada de innovación, al permitir a los profesionales utilizar la mayor parte de su tiempo en la resolución creativa de problemas y el pensamiento estratégico de alto nivel. Este enfoque revolucionario nos invita a reimaginar el desarrollo de software como una relación simbiótica entre el intelecto humano y la inteligencia artificial, donde el todo es mayor que la suma de sus partes. En este artículo examinaremos algunas de las herramientas de la Gen AI para entender qué impacto pueden tener en el desarrollo de software y cómo pueden dar lugar a nuevas formas de trabajo. Además, reflexionaremos sobre algunas de sus implicaciones legales y éticas, y su impacto en el Software Craftsmanship.

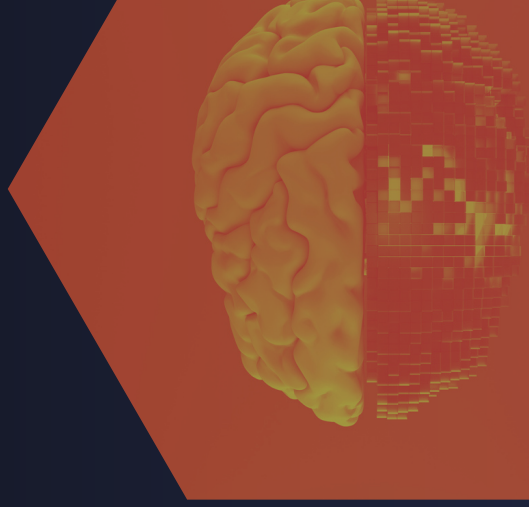
Qué es la IA Generativa

La IA generativa es un tipo de inteligencia artificial capaz de producir diversos formatos de contenido: texto, imágenes, audio y vídeo. Esto es posible gracias al uso de enormes cantidades de datos de entrenamiento procedentes de diversas fuentes, la detección de patrones y la creación de nuevos datos basados en estos patrones y de datos sintéticos a partir de un input dado y en respuesta a instrucciones.

Este arte de dar instrucciones precisas es un subcampo del data science y del machine learning también conocido como Prompt Engineering. Consiste en dar instrucciones claras y concisas para obtener los mejores resultados los resultados de las herramientas de Gen AI. De este modo, la asistencia será más inteligente y eficiente para diversas tareas.

Los modelos de IA generativa aprenden los patrones y la estructura de sus datos de entrenamiento y luego generan nuevos datos con características similares. La IA generativa tiene aplicaciones potenciales en una amplia gama de sectores, como el arte, el desarrollo de software, la sanidad, las finanzas, el marketing o la moda.

Hay que tener en cuenta que los modelos de Gen AI no son aplicables a todo ni representan la inteligencia general, ya que carecen de sentido común e inteligencia emocional y están diseñados únicamente para generar nuevos datos que se parezcan a un conjunto de datos determinado.



Chat GPT

El transformador generativo pre entrenado de OpenAI o ChatGPT, no es estrictamente una herramienta de desarrollo de software. Se trata más bien de una interfaz de usuario basada en un modelo lingüístico de aprendizaje automático capaz de generar textos en respuesta a las entradas que recibe. El modelo es tan amplio que se denomina Large Language Model (LLM).

Sin embargo, a pesar de no estar específicamente dirigido al desarrollo de software, ChatGPT ha influido en la forma en que los desarrolladores abordan su trabajo, ya que es capaz de generar fragmentos de código para solucionar diversos problemas de programación. Es importante señalar que las sugerencias que ofrece ChatGPT se generan a partir de la gran cantidad de datos que ha recopilado y sobre los que se ha formado, principalmente de Internet.

La iteración más reciente (en el momento de escribir este artículo) del modelo subyacente GPT4.5 ha incorporado mejoras a las respuestas que ofrece. Éstas pueden utilizarse para diversas tareas relacionadas con el desarrollo de software, como mejorar la documentación, generar casos de prueba, criticar el código y, por supuesto, generar fragmentos de código.

ChatGPT puede ayudar a comprender un lenguaje o framework, facilitando a los profesionales el acceso rápido a la información que necesitan. Asimismo ofrece sugerencias de código, lo que ahorra tiempo en tareas rutinarias, y ayuda en la detección y depuración de errores, proporcionando comentarios y posibles soluciones. También acelera la creación de prototipos y la exploración, permitiendo a los desarrolladores experimentar con diferentes ideas. Todo ello en un lenguaje accesible para usuarios con distintos niveles de programación.

En Codurance defendemos las prácticas de Extreme Programming (XP) en los proyectos con nuestros clientes. Una de ellas es la programación en parejas (pair programming). Sin embargo, adoptamos un enfoque pragmático al respecto: no todo requiere pair programming. Con la llegada de ChatGPT y herramientas similares, es posible imitar algunas de las ventajas de estas prácticas.

Aunque está claro que ChatGPT tiene mucho que ofrecer, también hay que tener en cuenta sus limitaciones. Puede carecer de comprensión contextual en función del plugin que se utilice, dando lugar a sugerencias inexactas que requieren validación humana. De igual forma, la dependencia del modelo de los datos de entrenamiento puede introducir sesgos o ejemplos de código defectuosos. También, es posible que no comprenda la intención del desarrollador o el contexto del proyecto y que no contemple plenamente la funcionalidad deseada.

A su vez, la privacidad y seguridad podrían verse afectadas en caso de que se comparta información sensible, y hay que prestar atención a consideraciones éticas, como evitar códigos discriminatorios. En general, depender en exceso de ChatGPT puede obstaculizar el desarrollo de habilidades, y la colaboración en tiempo real sigue siendo esencial para generar debates en profundidad.

GitHub Copilot

GitHub Copilot es una de las herramientas con mayor acogida en el sector del desarrollo de software. Nos sorprendió gratamente su velocidad de sugerencias y precisión, y ha funcionado muy bien junto con el desarrollo guiado por pruebas (TDD). Fue casi como tener a otra persona haciendo pairing con nosotros. Los fragmentos de código generados no siempre eran perfectos y a menudo había que retocarlos, pero nos permitió evitar el exceso de configuración.

Cuando un test-case está bien nombrado, Copilot intentará rellenar el cuerpo de la prueba basándose en el nombre de nuestro método y utilizando el contexto del código. Aunque proporcionar instrucciones paso a paso o dar indicaciones no es el uso típico de Copilot en codificación, funciona bien para crear el cuerpo de la prueba.

Además, la latencia de las sugerencias es bastante buena, aunque el desplazamiento por los fragmentos sugeridos no nos pareció tan natural como otras opciones.

Copilot está en constante desarrollo, con muchas características que se lanzan de forma regular. Una de las nuevas incorporaciones al plugin es la función de chat (sólo disponible en VSCode en el momento de escribir este artículo), así como los Copilot Labs, que tienen el concepto de pinceles. Por ejemplo Refactor, Cleanup, Readable, etc., cuyo objetivo es hacer el código más limpio y legible, o ayudar a refactorizarlo.

Otra característica interesante de Copilot Labs es que puedes traducir fragmentos de código de un lenguaje a otro, por ejemplo, de Java a Python. Es probable que en el futuro sea posible traducir una aplicación entera, lo cual sería útil si alguien quiere migrar su base de código a otro lenguaje, quizá como parte de un proceso más amplio para evolucionar su pila técnica. Además, sería especialmente beneficioso para empresas que quieran emprender una iniciativa de modernización de software que implique bases de código o lenguajes más antiguos.

AWS Code Whisperer

Amazon AWS ha estado trabajando en su propio asistente de IA para el desarrollo de software. Similar a Copilot soporta muchos lenguajes, IDEs populares y cubre las mismas necesidades de generación de código asistido por la IA. Al igual que en nuestro enfoque de TDD con Copilot, pudimos ejecutar métodos a partir de un caso de prueba bien nombrado.

No se conocen bien los detalles de implementación de CodeWhisperer, pero se ha desarrollado en código abierto o interno de Amazon. Actualmente se ofrece de forma gratuita para usuarios con una cuenta de AWS Builders y una versión con licencia, llamada CodeWhisperer Professional, para organizaciones que utilizan los servicios en la nube de AWS.

Es importante señalar que, al igual que Copilot, CodeWhisperer también puede ayudar a escribir aplicaciones. Sin embargo, un área en la que CodeWhisperer realmente destaca es en la escritura de código para los servicios de AWS como AWS S3, Lambda, etc. Copilot también cumple con esta función, pero CodeWhisperer hace un mejor trabajo.

Otra característica de Code Whisperer es que comprueba si el código generado contiene vulnerabilidades de seguridad incluidas en el Top 10 de OWASP. Una vez identificada una amenaza, CodeWhisperer proporcionará sugerencias sobre cómo solucionarla. Sin embargo, esta funcionalidad está limitada a Python, Java y JavaScript por el momento.

En general, creemos que CodeWhisperer en su estado actual no es tan rico en funciones como Copilot, pero esto podría cambiar en el futuro, ya que es un ámbito en rápida evolución. Nosotros continuaremos monitoreando ambas herramientas activamente.

Privacidad y Seguridad

La privacidad y la seguridad son aspectos cruciales a tener en cuenta cuando se utilizan herramientas como GitHub Copilot y CodeWhisperer. Aunque estos asistentes pretenden ofrecer sugerencias eficientes y precisas e intentan filtrar las amenazas de sus datos, existe la posibilidad de que generen código con vulnerabilidades de seguridad.

Hay que reconocer que la llegada de estas herramientas de IA conlleva implicaciones de seguridad que es preciso abordar. A medida que estas herramientas se vuelven más sofisticadas, se hace imperativo contar con medidas para salvaguardar la información sensible y evitar posibles usos indebidos.

Como usuarios, es importante recordar que somos responsables del código generado, ya que las respuestas se adaptan a nuestros requisitos específicos en función de las indicaciones que les damos.

Tanto GitHub Copilot como CodeWhisperer recopilan datos para mejorar sus servicios. GitHub Copilot captura avisos en tiempo real, sugerencias generadas por IA y datos de participación del usuario dentro del IDE. Sin embargo, GitHub Copilot for Business no almacena avisos ni sugerencias, sólo datos pseudónimos sobre la participación del usuario.

De forma similar, CodeWhisperer (individual) almacena datos de uso y contenido para mejorar su servicio, pero los usuarios pueden elegir no compartirlos. Los datos recopilados incluyen telemetría y contenido del cliente, aunque en el modo profesional la recopilación de contenido está desactivada. La telemetría no incluye código real ni información personal identificable (PII), y el uso de los datos puede limitarse a la propia VPC para un mejor control.

Aunque ambas herramientas carecen de acceso al código fuente, sí pueden tener acceso a información contextual como la pila tecnológica que se utiliza y las características del dominio. Para evitar cualquier problema de propiedad intelectual, se recomienda consultar con asesores jurídicos. Entender estas consideraciones nos ayuda a tomar decisiones contrastadas y a utilizar estas potentes herramientas salvaguardando nuestro código e información personal.

IA y Software Craftsmanship

Quienes conozcan a Codurance sabrán que los principios del Software Craftsmanship son la base de nuestro trabajo. Por lo tanto, consideramos necesario analizar cómo la IA puede afectar a este enfoque. Para aquellos que no estén familiarizados con el Software Craftsmanship, es una filosofía que busca elevar el nivel de calidad en la industria del software a través del profesionalismo y la excelencia técnica.

Creemos que incluso con el auge de la IA y las herramientas de asistencia al código, el Software Craftsmanship sigue ocupando un lugar importante en el desarrollo de software. Aunque las herramientas de Gen AI pueden generar código funcionalmente correcto, es preciso recordar que no son más que herramientas. Como tales, carecen de comprensión sobre los matices del ámbito empresarial, las necesidades específicas de un usuario o el contexto más amplio en el que se encuentra el software.

El Software Craftsmanship no busca sólo crear software que funcione, sino que esté bien diseñado. Es decir, producir soluciones robustas, fiables y escalables que nos permitan sentirnos orgullosos del resultado que creamos. Este sentimiento debe vivir en cada craftsperson y no puede externalizarse a una herramienta.

Sin embargo, creemos que el Software Craftsmanship y la IA pueden coexistir y complementarse. Por ejemplo, la Gen AI y otras herramientas de asistencia al código pueden utilizarse para automatizar tareas repetitivas que no sean muy complicadas pero que requieran mucho esfuerzo. Esto permite a los desarrolladores dedicar más tiempo y energía a actividades que generen valor añadido para el negocio, como el diseño arquitectónico, la resolución de problemas complejos y la implementación de funciones críticas.

Conclusión

Al principio de este artículo planteamos la cuestión de si la IA está preparada para revolucionar el desarrollo de software. Tras analizarlo, consideramos que varias herramientas de Gen AI tienen ciertamente el potencial de aumentar la productividad, entre otras cosas, pero que aún queda mucho camino por recorrer.

Quienes sigan de cerca la IA sabrán que se trata de un campo en constante evolución y desarrollo, en el que cada semana aparecen nuevas actualizaciones. Los modelos de IA se entrenan y se lanzan nuevas versiones, cada una aparentemente más avanzada que la anterior. En el momento de escribir estas líneas, Meta acababa de lanzar su modelo de código abierto de segunda generación Llama2. Todo esto significa que, aunque la IA no está revolucionando el desarrollo de software por el momento, tiene el potencial de transformarlo.

Lo que está claro es que la IA ha llegado para quedarse y es vital que los líderes en tecnología evalúen cómo puede afectar a sus procesos para sacarle el máximo beneficio. Además, se deben tener en cuenta las implicaciones legales y de seguridad comentadas anteriormente, así como la forma en que las prácticas actuales pueden evolucionar y optimizarse. Por ejemplo, un tema común de debate en los últimos tiempos es cómo la IA puede desempeñar un papel en el TDD.

Es importante comprender todas las implicaciones de adoptar herramientas de IA en tu proceso de desarrollo de software. Aunque pueden ofrecer una ayuda enorme, los resultados que generan son responsabilidad de cada empresa y esto plantea problemas de propiedad intelectual. Los desarrolladores y responsables técnicos deben ser conscientes de ello y tomar las precauciones necesarias para proteger sus propias ideas y contribuciones. Dicho esto, el último modelo de IA de Meta es sin duda un avance interesante en este ámbito.

Además de comprender las implicaciones legales y éticas, también es necesario tener en cuenta que mayor cantidad de código no significa mayor calidad; de hecho, es posible observar mucho código inadecuado cuando se ha confiado demasiado en los primeros modelos de IA. Debemos ser cautos al utilizar las herramientas de apoyo al código de la Gen AI y aplicar el mismo nivel de cuidado y atención al código que generan como al que escribimos.

En Codurance somos prudentemente optimistas ante el potencial de la IA en el desarrollo de software. Nos entusiasman las posibilidades que la IA podría abrir y las mejoras de productividad que podría aportar a la forma de desarrollar software en el futuro. Pero, al mismo tiempo, somos conscientes de las implicaciones éticas, legales y de seguridad.

Sobre los autores

Matt Belcher
Principal Craftsperson

Matt tiene más de 15 años de experiencia en el desarrollo de software, principalmente como consultor. Ha trabajado con un gran número de clientes en muchos países, tech stacks, lenguajes y arquitecturas técnicas. A lo largo de su carrera ha desempeñado numerosas funciones, desde desarrollador de software hasta arquitecto de soluciones. Hoy en día Matt trabaja junto a líderes técnicos de varias organizaciones asesorando y apoyando en áreas como arquitectura técnica, desarrollo de software y prácticas ágiles.

Le apasiona ayudar a las empresas a mejorar sus capacidades de entrega de software, ya sea incorporando las mejores prácticas o estableciendo una visión técnica clara. Además, le interesan especialmente las arquitecturas basadas en eventos y serverless.

Danish Javed
Software Craftsperson

Como desarrollador backend experimentado y coorganizador de Software Crafters North, a Danish le apasiona ofrecer soluciones de software de buena calidad. Ha perfeccionado sus habilidades en la creación y el mantenimiento de aplicaciones web de alto rendimiento que son robustas y escalables. Cree firmemente en el poder de TDD, la arquitectura limpia y los principios de código limpio para crear las mejores soluciones para los clientes.

Danish se desenvuelve a gusto en entornos de rápido crecimiento y alta presión, y está comprometido con la entrega de resultados de calidad. También es un miembro activo de la comunidad de desarrollo de software y le encanta compartir sus experiencias para ayudar a otros a crecer. Ya sea coorganizando eventos, como en la Software Crafters Manchester o manteniéndose al día de los últimos avances del sector, Danish siempre está buscando formas de superarse a sí mismo.

@codurance_ES **hello@codurance.com** **www.codurance.com/es**