INESDI + BUSINESS TECHSCHOOL

+ Descubre cómo las herramientas de IA generativa están cambiando la forma en que desarrollamos software

IA GENERATIVA Y SU IMPACTO EN EL DESARROLLO DE SOFTWARE

Autor
Jaume Puigbó

RED DE EDUCACIÓN SUPERIOR
Planeta Formación y Universidades

INESDI BUSINESS TECHSCHOOL

Campus Barcelona:

Av. Gran Via de l'Hospitalet, 153,
08908 L'Hospitalet de Llobregat,
Barcelona

Tel: 934 457 903

Campus Madrid:

C/ de Arapiles, 14,
28015 Madrid

Tel: 919 03 22 01

Campus Online:

Desde cualquier lugar
del mundo

info@inesdi.com | www.inesdi.com

ÍNDICE

1	La IA Generativa y su impacto en el desarrollo de software	5
	1.1 Introducción	6
	1.2 Generando código	7
	1.3 Panorama actual	9
2	Chatbots como asistentes: escribe más, programa menos	10
	2.1 Resolución de problemas técnicos	13
	2.2 Documentación de código	14
	2.3 Creación de prototipos rápidos	14
	2.4 Traducción de código	14
	2.5 Un caso de éxito: las soluciones text-to-SQL	15
	2.6 Ventajas del uso de LLM como asistentes	16
	2.7 Riesgos	16

3 Desarrollo asistido por IA. Paradigmas emergentes 18
3.1 Desarrollo asistido por IA 19
3.2 Aprendizaje automático en el ciclo de desarrollo 20
3.3 Automatización de pruebas 20
3.4 Mantenimiento predictivo 21
3.5 Generación automática de documentación 21
3.6 Personalización de experiencias de desarrollo 21
3.7 Cómo funciona el 'infilling' 23
3.8 Modelos 'long context' 24
4 Modelos LLM específicos con entrenamiento propio 26
4.1 Entrenamiento desde modelos base 27
4.2 Cambios de responsabilidades en líderes de equipos de desarrollo 30
4.3 Retos relacionados con la privacidad y la seguridad 33
5 Conclusiones 35
5.1 Bonus track: herramientas de IA para el desarrollo de software 39
6 Bibliografía y Referencias 44

CAP. 1

LA IA GENERATIVA Y SU IMPACTO EN EL DESARROLLO DE SOFTWARE

1.1 Introducción

La Inteligencia Artificial (IA) ha estado revolucionando diversas industrias, y el desarrollo de software no es una excepción. Las herramientas de IA generativa están cambiando la forma en que desarrollamos software y están afectando a los diferentes roles dentro del ciclo de desarrollo. Entre muchas otras tendencias dentro del mundo de la tecnología ya se apuntaba al desarrollo asistido por IA como una de las que más iban a impactar en el sector. Con cierta prudencia de momento, pero con una evolución clara y acelerada.

¿Cómo están cambiando los fundamentos de un trabajo de décadas de antigüedad? ¿Ayudarán a los programadores a crear más y mejor software? ¿O se verán atascados en luchas legales sobre la privacidad y la propiedad intelectual? ¿Vamos a asistir al famoso mito de la desaparición del programador?

La distopía radica en discernir entre ver a la IA generativa como una herramienta o como una entidad autónoma.

El objetivo de este informe es explorar el impacto de la IA generativa en el desarrollo de software, considerando tanto sus beneficios como sus desafíos. A medida que esta tecnología continúa evolucionando, es crucial comprender cómo está cambiando las dinámicas del desarrollo y qué implicaciones tiene para los desarrolladores, las empresas y el mercado en general. Este estudio se centrará en varios aspectos clave.

1.2 Generando código

A primera vista, programar o escribir código implica escribir declaraciones e instrucciones en algún lenguaje de programación en un archivo de texto. Esto luego se traduce en código de máquina[1] que una computadora puede ejecutar, un nivel superior a los 0 y 1 del binario básico computacional. El código es una secuencia de instrucciones que un programador escribe para decirle a un dispositivo (como una computadora) qué hacer.

Los lenguajes de programación clásicos, formales y deterministas existen por una razón. Permiten que los programas especifiquen con precisión lo que se requiere al dispositivo al que van dirigidos. En los inicios de los tiempos de la informática se programaba en lenguaje máquina para luego ir evolucionando entre distintas generaciones de lenguajes de programación cada vez más distantes de esa arcana codificación inicial y más cercanos al lenguaje natural.

Conocer estos lenguajes, codificar con ellos, es una síntesis de aplicación de lógica y algoritmia con unas determinadas reglas gramaticales.

Pero el reto de programar "en inglés" como comentaremos más adelante es cada vez más suculento desde la aparición de los modelos de IA generativa, a pesar de que todavía tiene camino por recorrer.

Utópicamente, para muchos desarrolladores, programar es una de las habilidades más útiles y poderosas que se pueden aprender para lograr que tu visión se convierta en realidad. Es mucho más que una habilidad técnica, es una pasión que los impulsa a desafiar los límites de lo posible y a imaginar un mundo mejor a través de la tecnología. Gracias a su dominio de cada una de las disciplinas asociadas, pueden desarrollar soluciones innovadoras y disruptivas que generan un impacto positivo en la sociedad y en lo personal.

[1] El lenguaje de máquina es un sistema de códigos directamente interpretable por el microprocesador de una computadora, específico para cada arquitectura y ejecutable sin traducción adicional. La línea evolutiva sería: lenguaje máquina, lenguaje ensamblador y lenguaje de alto nivel. Actualmente se está desarrollando en lenguajes de alto nivel de tercera y cuarta generación (C, C++, Java, Python, Javascript, etc).

Cada lenguaje de programación tiene su propio conjunto de reglas que determinan si una línea de código es válida o no. Debido a esto, el código que se escribe en un lenguaje de programación será diferente al código que se escriba en otros lenguajes de programación. Ello implica un proceso de aprendizaje y especialización que requiere dedicación y esfuerzo.

Hoy en día, en la práctica, los programadores también pasan mucho tiempo escudriñando en Google, buscando soluciones para problemas comunes u hojeando foros en línea para encontrar formas más rápidas de escribir un algoritmo en un lenguaje en concreto. Los fragmentos existentes de código prescrito se reutilizan y el nuevo software a menudo se une como un collage.

Para las empresas que buscan innovar y mantenerse competitivas en el mercado del software actual, la integración de la inteligencia artificial en el desarrollo de software se ha convertido en un componente crítico.

Uno de los principales beneficios de la IA generativa en el desarrollo de software es su capacidad para aumentar la productividad. Los desarrolladores pueden utilizar asistentes de IA para generar código, detectar errores y realizar pruebas automatizadas, lo que reduce significativamente el tiempo y el esfuerzo necesarios para completar proyectos. Además, estas herramientas pueden adaptarse a diferentes lenguajes y entornos de programación, proporcionando soluciones personalizadas que se ajustan a las necesidades específicas de cada proyecto. Esto no sólo acelera el proceso de desarrollo, sino que también puede mejorar la calidad del software producido.

1.3 Panorama actual

Desde el advenimiento de ChatGPT en noviembre de 2022 muchos programadores a título personal han ido explorando vías de prospección de las posibilidades de la inteligencia artificial generativa en el desarrollo de software.

Aunque solo tratemos algunas de ellas, en estos momentos tenemos un ecosistema en ebullición de herramientas. Éstas no están arrinconando al desarrollador tradicional, sino que le están dotando de una potencia nunca antes vista.

El ecosistema de IA en el desarrollo de software incluye una variedad de herramientas y tecnologías que ayudan en diferentes etapas del desarrollo.

Podríamos definir tres grandes fases de adopción:

1. En el plano más elemental y en su forma más básica, consiste en la utilización de los chatbots como asistentes (Bard, ChatGPT, Claude, Gemini,...) , los cuales juegan un papel importante como herramientas que complementan y enriquecen nuestras capacidades.

2. Herramientas de desarrollo asistido por IA. Son herramientas específicas ya sea en plataformas (GitHub, GitLab, Bitbucket, Azure DevOps, Visual Studio Code,...) o como complementos integrados en los IDE (entorno de desarrollo integrado) más utilizados (Eclipse, Microsoft Visual Studio, BlueJ,NetBeans, IntelliJ IDEA). Son herramientas que sugieren código mientras se escribe y que pueden interactuar con el programador a través de prompts (que se insertan en el código como comentarios).

3. Los modelos específicos con entrenamiento propio para el desarrollo de software implican un avance para su uso a modo de herramienta autónoma a partir de la especialización en un lenguaje o un entorno en el que se les proporciona una amplia base de conocimiento específico.

CAP. 2

CHATBOTS COMO ASISTENTES: ESCRIBE MÁS, PROGRAMA MENOS

Bajo este lema, diversos autores han escrito infinidad de artículos sobre los beneficios y la conveniencia del uso de modelos LLM (Large Language Models) como asistentes de desarrollo de software, cuyo punto culminante fue el post en X de Andrej Karpathy "El nuevo lenguaje de programación más popular es el inglés".

Este es un interesante concepto que fue inicialmente explorado por Charles Simonyi con su empresa Intentional Software, que buscaba permitir a los desarrolladores especificar lo que querían sin escribir código complejo. Aunque la empresa no tuvo el éxito esperado, la idea de programar en inglés (en lenguaje natural) sigue siendo atractiva, especialmente con el avance de los modelos de lenguaje como los LLMs.

Poder expresarse en 'plain English' con un asistente que nos acompañe en nuestros avatares durante la codificación de programas para aplicaciones puede resultar muy satisfactorio. Suena un tanto distópico, pero ya es posible, por lo menos hasta cierto nivel.

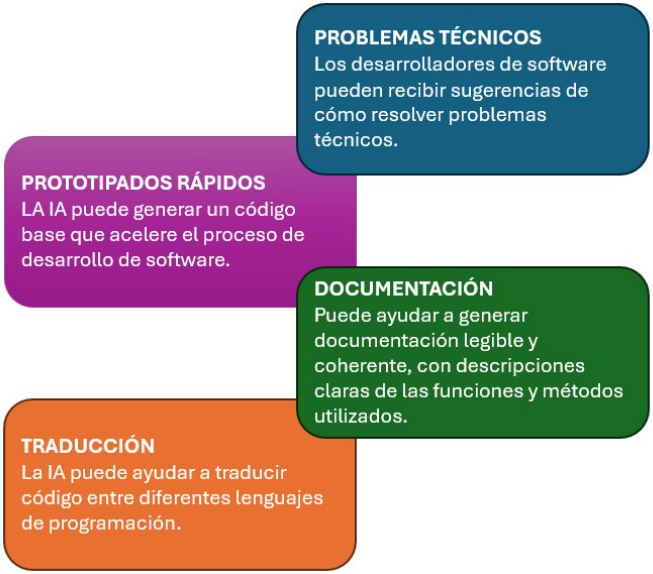
Inicialmente, muchos programadores (y no programadores) han ido adoptando el uso de la IA generativa a modo de asistente y algunos de los casos de uso más destacados incluyen:

- **Generación de código:** Generar fragmentos de código o funciones completas según los requisitos o especificaciones dados.

- **Asistencia de sintaxis y API:** Sugerir fragmentos de código y ayuda con la sintaxis de programación.
- **Generación de Ejemplos y Plantillas:** La IA generativa puede proporcionar ejemplos y plantillas de código que los no programadores pueden utilizar como punto de partida, lo que les permite entender mejor cómo se estructura y se desarrolla el código
- **Soporte para depuración:** Identificar problemas y sugerir soluciones para arreglar código.
- **Guía de algoritmos y estructuras:** Ayuda a seleccionar el algoritmo o estructuras de datos adecuados.
- **Recursos de aprendizaje:** Recomendar tutoriales y documentación para expandir el conocimiento en programación.
- **Revisión de código:** Analizar y revisar código para mejores prácticas y optimizaciones.
- **Generación de documentación** a partir de código fuente.
- **Tareas de procesamiento de lenguaje natural (NLP):** Asistir en tareas de NLP, como preprocesamiento de texto o extraer datos de información no estructurada.
- **Resolución de problemas:** Desarrollar soluciones para problemas propuestos complejos. ●
- **Preguntas de programación generales:** Responde preguntas de programación y principios de ingeniería de software.

2.1 Resolución de problemas técnicos

En la actualidad, el uso de chatbots basados en inteligencia artificial, como ChatGPT, Claude, Copilot o Gemini, ha revolucionado la forma en que se desarrolla software. Estos chatbots proporcionan respuestas rápidas a problemas técnicos y errores de código, brindando a los desarrolladores explicaciones detalladas y sugerencias sobre cómo abordar un problema específico. Más allá de sus funciones relacionadas con el código, estas herramientas se convierten en una valiosa fuente de conocimiento para los profesionales del desarrollo, comparable a un motor de búsqueda o a un sitio web de preguntas y respuestas como Stack Overflow. En resumen, estos chatbots son capaces de comprender conceptos complejos de programación y proporcionar información relevante, simplificando así la tarea de los desarrolladores al buscar respuestas rápidas a preguntas específicas interactuando con el modelo.



2.2 Documentación de código

Es frecuente encontrarse con código que carece de una buena documentación, lo que dificulta su comprensión y mantenimiento a largo plazo. Sin embargo, la inteligencia artificial generativa puede crear documentación detallada y precisa para los requisitos del código, aprovechando sus habilidades en procesamiento del lenguaje natural. Este enfoque ahorra un tiempo considerable a los desarrolladores, quienes suelen invertir esfuerzos significativos en la creación de documentación exhaustiva.

2.3 Creación de prototipos rápidos

Los desarrolladores pueden utilizar la IA para crear prototipos rápidos de nuevas funciones o características. Al describir lo que desean, ChatGPT, Gemini y compañía pueden ayudar a generar la estructura de un código base que acelere el proceso de desarrollo de software.

2.4 Traducción de código

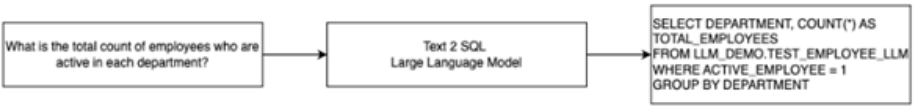
Los chatbots de IA pueden ayudar a traducir código entre diferentes lenguajes de programación. Esto es útil cuando los desarrolladores trabajan en proyectos que requieren conocimientos en varios lenguajes de programación.

2.5 Un caso de éxito: las soluciones text-to-SQL

Con la aparición de grandes modelos de lenguaje (LLM), la generación de SQL basada en lenguaje natural (NLP) ha experimentado una transformación significativa. Al demostrar un rendimiento excepcional, los LLM ahora son capaces de generar consultas SQL precisas a partir de descripciones en lenguaje natural.

Sin embargo, todavía quedan desafíos. En primer lugar, el lenguaje humano es inherentemente ambiguo y depende del contexto, mientras que SQL es preciso, matemático y estructurado. Esta brecha puede resultar en una conversión inexacta de las necesidades del usuario al SQL que se genera.

En segundo lugar, es posible que necesite crear funciones de texto a SQL para cada base de datos porque los datos a menudo no se almacenan en un único destino. Es posible que deba volver a crear la capacidad de cada base de datos para permitir a los usuarios la generación de SQL basada en NLP.



En tercer lugar, a pesar de la mayor adopción de soluciones de análisis centralizadas, como data-lakes y otros almacenes de datos, la complejidad aumenta con los diferentes nombres de tablas y otros metadatos que se requieren para crear el SQL para las fuentes deseadas. Por lo tanto, la recopilación de metadatos completos y de alta calidad todavía sigue siendo un desafío.

2.6 Ventajas del uso de LLM como asistentes

-
- Asistencia rápida para resolver problemas (¿y StackOverflow? Esto requeriría un artículo aparte).
- Exploración de soluciones: puede ayudar a explorar diferentes enfoques y soluciones, brindando ideas y perspectivas adicionales.
- Aumento de velocidad en tareas como traducción entre lenguajes, mejora y refactorización de código.

2.7 Riesgos

- Puede generar resultados plausibles, pero completamente incorrectos o inventados. Las temidas alucinaciones podrían revertir en sugerencias incorrectas que pueden hacer perder más tiempo.
- Generación de código incorrecto o inseguro, lo que podría introducir errores o vulnerabilidades en las aplicaciones.
- Conocimiento limitado del contexto específico de un proyecto o librería.
- Dependencia excesiva en el modelo por parte de los desarrolladores.
- No sirve para desarrollos innovadores o que por ejemplo usen nuevas librerías. No conoce lo nuevo, solo hasta la fecha de los datos con los que fue entrenado.
- Generación de código no defensivo. Se deben añadir prompts adicionales para comprobación de errores o parámetros.

- Sugerencias incorrectas que pueden hacer perder más tiempo.
- Parte del tiempo ganado en la generación se pierde en la revisión.

Estos casos de uso demuestran cómo la IA generativa puede ser una herramienta poderosa en el desarrollo de software, acelerando el proceso en todo el ciclo de desarrollo, mejorando la calidad del código y ofreciendo nuevas formas de interactuar con las aplicaciones y servicios. Sin embargo, es importante recordar que, si bien Chat GPT, Gemini o Claude son unas herramientas valiosas, no reemplazan la experiencia y el juicio humano en el desarrollo de software.

Sin embargo, la adopción de la IA generativa no está exenta de desafíos. Uno de los principales problemas es la dependencia excesiva de la tecnología, que puede llevar a una disminución en la habilidad y creatividad de los desarrolladores. Además, existen preocupaciones sobre la seguridad y la privacidad, ya que los modelos de IA pueden ser vulnerables a ataques y manipulaciones. Es esencial que las empresas implementen medidas de seguridad robustas y mantengan un equilibrio entre el uso de IA y la intervención humana para mitigar estos riesgos.

CAP. 3

DESARROLLO ASISTIDO POR IA: PARADIGMAS EMERGENTES

En el campo del desarrollo de software, los paradigmas emergentes están siendo fuertemente influenciados por la inteligencia artificial (IA) y el aprendizaje automático (AA). Estas tecnologías están redefiniendo las prácticas y herramientas tradicionales, ofreciendo nuevas maneras de abordar el ciclo de vida del desarrollo de software. La clave se centra en emplearla en base a todo el repositorio de desarrollo como una entidad autónoma preentrenada.

Estas herramientas están en boga y gozan de una popularidad sin precedentes en el entorno profesional de desarrollo. La integración con interfaces de lenguaje natural para recibir instrucciones complejas representa un avance significativo en el campo de la programación, facilitando la comunicación entre los desarrolladores y las máquinas de una manera más intuitiva y eficiente.

3.1 Desarrollo asistido por IA

El desarrollo asistido por IA está revolucionando la forma en que los programadores escriben y depuran código. Herramientas como GitHub Copilot y Tabnine utilizan modelos de lenguaje avanzados para sugerir líneas de código, completaciones automáticas y soluciones a problemas comunes. Estas herramientas no solo aumentan la productividad de los desarrolladores, sino que también mejoran la calidad del código al proporcionar soluciones optimizadas y más seguras.

3.2 Aprendizaje automático en el ciclo de desarrollo

El aprendizaje automático se integra en el ciclo de desarrollo de software para ofrecer análisis predictivos y automatización en tareas repetitivas. Algoritmos de AA pueden predecir posibles errores, optimizar procesos de prueba y personalizar experiencias de usuario. Esta integración permite a los equipos de desarrollo centrarse en tareas más creativas y estratégicas, mientras la IA se encarga de las operaciones rutinarias.

3.3 Automatización de pruebas

Uno de los mayores beneficios del aprendizaje automático en el ciclo de desarrollo es la automatización de pruebas. Las herramientas de AA (aprendizaje automático) pueden generar y ejecutar casos de prueba basados en patrones de uso anteriores, identificando rápidamente errores y asegurando que el software cumpla con los estándares de calidad. Esta capacidad reduce significativamente el tiempo y el esfuerzo necesarios para realizar pruebas exhaustivas, acelerando el tiempo de disponibilización del producto.

Aprovechar la IA en todo el ciclo de vida de las pruebas de software con la IA aumentada puede proporcionar beneficios en distintas áreas: ●

- Planificación y priorización de pruebas
- Creación y mantenimiento de pruebas
- Generación de datos de prueba
- Análisis de pruebas y defectos

3.4 Mantenimiento predictivo

La IA y el aprendizaje automático también desempeñan un papel crucial en el mantenimiento predictivo. Analizando datos históricos y patrones de uso, estas tecnologías pueden anticipar cuándo es probable que ocurra una falla en el software y sugerir acciones preventivas. Este enfoque proactivo minimiza el tiempo de inactividad y asegura una experiencia de usuario más fluida y confiable.

3.5 Generación automática de documentación

La generación automática de documentación es otro avance significativo facilitado por la IA. Los asistentes de código pueden generar comentarios detallados y documentación basada en el análisis del código fuente, asegurando que cada componente esté claramente descrito. Esto es esencial para la colaboración en equipos grandes y para el mantenimiento a largo plazo del software.

3.6 Personalización de experiencias de desarrollo

Finalmente, la IA y el aprendizaje automático permiten la personalización de experiencias de desarrollo. Analizando los hábitos y preferencias de los desarrolladores, estas herramientas pueden ajustar su comportamiento para

proporcionar una asistencia más relevante y contextualmente adecuada. Esto no solo mejora la eficiencia del desarrollo, sino que también crea un entorno de trabajo más intuitivo y agradable.

En resumen, los paradigmas emergentes de desarrollo asistido por IA y aprendizaje automático en el ciclo de desarrollo están transformando la industria del software. Estas tecnologías no solo optimizan y aceleran el proceso de desarrollo, sino que también mejoran la calidad del software y la experiencia del usuario. A medida que la IA continúa evolucionando, su integración en las plataformas como GitHub Copilot, por ejemplo, seguirá abriendo nuevas posibilidades y redefiniendo las prácticas tradicionales.

Con la ayuda de las herramientas de codificación de IA, puede desarrollar aplicaciones más rápidamente automatizando las tareas de codificación repetitivas y aprovechando la finalización del código. Entre sus características principales se cuentan:

- **Modelos más grandes y eficientes:** Con billones de parámetros, los LLM tienen una capacidad muy superior a la de modelos anteriores para capturar los matices del lenguaje de programación analizando todo un repositorio de desarrollos.
- **Mejor generalización:** Al entrenarse en modelos de LLM como punto de partida, heredan conocimientos amplios sobre lenguaje natural y patrones de programación. Esto les permite generalizar bien en nuevos contextos y lenguajes.
- **Entendimiento multifacético:** Muestra un entendimiento integral de código, documentación y lenguaje natural. Pueden conectar estos elementos para tareas como generación automatizada de documentación.
- **Capacidad de infilling única:** La habilidad para rellenar secciones faltantes de código basado en el contexto circundante, infilling, es un paso adelante respecto a modelos anteriores.

3.7 Cómo funciona el *"infiling"*

El infilling, en el contexto de la programación y el procesamiento del lenguaje natural, se refiere a la capacidad de un modelo para identificar y completar partes faltantes o incompletas de un programa o texto basándose en el contexto circundante. Es crucial para:

- **Mejora de la productividad:** Ayuda a los programadores a completar rápidamente fragmentos de código, reduciendo el tiempo de desarrollo y depuración.
- **Autocompletado inteligente:** Ofrece sugerencias más relevantes y precisas al considerar el contexto completo.
- **Corrección y depuración:** Facilita la corrección de códigos, especialmente en casos donde faltan partes esenciales.

Aplicaciones prácticas del infillig:

- **IDEs y editores de código:** Mejora la eficiencia ofreciendo su capacidad de autocompletado inteligente en las herramientas donde el programador codifica.
- **Generación de documentación:** Completa automáticamente comentarios y documentación.
- **Formación y educación:** Ayuda a los estudiantes y nuevos desarrolladores a identificar y completar partes faltantes en ejercicios de programación.

3.8 Modelos *"long context"*

Los long context de entrada (extenso contexto de entrada) son esenciales en el procesamiento del lenguaje natural (PLN) y la generación de código, ya que permiten a los modelos de lenguaje procesar y comprender secuencias extensas de texto o código.

Definición de long context de entrada:

- Un "contexto" es la secuencia de tokens (palabras, caracteres o símbolos) que el modelo utiliza como entrada.
- "Extenso" se refiere a la longitud de esta secuencia, que puede abarcar desde miles hasta decenas de miles de tokens.

Poder trabajar con estas extensas secuencias de código permite:

- **Razonamiento a nivel de repositorio:** Permite a los programadores trabajar con grandes bases de código que incluyen múltiples archivos y funciones.
- **Integración de información:** Facilita la combinación de información de diferentes partes de un texto o código para generar respuestas más coherentes.
- **Complejidad y profundidad:** Ayuda a abordar problemas que requieren una comprensión detallada de grandes cantidades de información.

Esto supone unas ventajas significativas:

- **Autocompletado de código:** Sugerir líneas de código relevantes en tiempo real mientras el programador escribe (infilling aumentado).
- **Corrección de errores:** Detectar y corregir errores sintácticos y lógicos en el código existente.
- **Generación de funciones:** Generar bloques completos de código para funciones específicas según la descripción del programador.

- **Documentación:** Generar comentarios y documentación técnica coherente basada en el código.
- **Migración de código:** Traducir código de un lenguaje a otro, como de C# a Python o Javascript.
- **Refactorización:** Sugerir formas de reestructurar y optimizar código heredado.

Estas herramientas están al alza. Un ejemplo, según el blog de GitHub, Copilot permite a más de 1,2 millones de desarrolladores codificar un 55 % más rápido, y que, en la actualidad, el 46 % del código está creado utilizando la herramienta, aumentando este porcentaje hasta el 61 % cuando hablamos de Java. Además, el 90 % de sus usuarios dicen completar las tareas más rápido gracias a Copilot, al 73 % le ayuda a mantener la fluidez mental y hasta el 75 % se siente más satisfecho con el trabajo realizado.

Hasta aquí los números. La idea clave detrás de GitHub Copilot y otras aplicaciones similares, a veces llamados asistentes de código, es colocar la información que los programadores necesitan justo al lado del código que están escribiendo. La herramienta rastrea el código y los comentarios (descripciones o notas escritas en lenguaje natural) en el archivo en el que está trabajando un programador, así como otros archivos a los que vincula o que han sido editados en el mismo proyecto, y envía todo este texto al modelo de lenguaje grande detrás de Copilot como indicación (GitHub desarrolló conjuntamente el modelo de Copilot, llamado Codex, con OpenAI. Es un modelo LLM ajustado al código). Luego, Copilot predice lo que el programador está tratando de hacer y sugiere código para hacerlo.

CAP. 4

MODELOS LLM ESPECÍFICOS CON ENTRENAMIENTO PROPIO

La opción más avanzada en cuanto a herramientas que asistan a la generación de código sería la de entrenar nuestro propio modelo LLM con datos específicos de nuestros programas y repositorios de código, con los lenguajes con los que desarrollamos y el paradigma de programación que usamos en nuestras aplicaciones.

Es una solución in-house que hay que calibrar bien, hay que escoger un buen modelo base, hay que contar con el coste de reentreno de dicho modelo y tener en cuenta la infraestructura, tanto por la capacidad de cómputo para el reentreno como por el almacenamiento y disponibilización del modelo y su continua actualización.

Almacenar localmente un modelo LLM de código abierto que haya sido reentrenado puede otorgar un mayor nivel de control y seguridad a las áreas de desarrollo de la organización.

4.1 Entrenamiento desde modelos base

La generación de modelos especializados requiere un punto de partida robusto y versátil. En lugar de comenzar desde cero o depender exclusivamente de conjuntos de datos especializados, utilizan una estrategia de inicialización basada en modelos LLM preexistentes (GPT para Copilot, LLaMa2 para Code Llama, etc).

Algunas **áreas de aplicación prometedoras** incluyen:

- Modelos especializados en lenguajes de programación específicos como Javascript, Python o Ruby.
- Mejor manejo de programación orientada a objetos y patrones avanzados para los cuales se puede entrenar nuestro modelo.

- Capacidades más fuertes de razonamiento lógico sobre código.
- Integración con interfaces de lenguaje natural para recibir instrucciones complejas.
- Generación de código completo para aplicaciones full-stack (frameworks de creación de páginas web, front-ends, etc).

¿Por qué iniciar un modelo propio desde un modelo base?

- **Amplia base de conocimiento:** Los modelos LLM ya han sido entrenados en grandes cantidades de texto y código, proporcionando una comprensión sólida del lenguaje natural y las estructuras de programación. Al empezar con un LLM, el nuevo modelo no tiene que aprender desde cero, lo que ahorra tiempo y recursos computacionales.
- **Flexibilidad:** Además de entrenar los modelos con código, éstos heredan habilidades de procesamiento de lenguaje natural y de código. Esto los hace más versátiles para tareas que involucran interacciones en lenguaje natural y código.
- **Mejor rendimiento comparativo:** Ciertos estudios han mostrado que iniciar desde un modelo base como Llama 2, Bard, Falcon o GPT y afinarlo para tareas específicas puede resultar en un mejor rendimiento en comparación con entrenar un modelo desde cero solo con datos especializados para una tarea dada.

Este enfoque de iniciar desde un modelo base y afinarlo con datos específicos es una estrategia poderosa que combina lo mejor de ambos mundos: la amplia comprensión del lenguaje natural y el conocimiento especializado del dominio del código.

Hoy en día, el espacio LLM de código abierto se está expandiendo rápidamente. En la actualidad, hay muchos más LLM de código abierto que propietarios, y es posible que pronto se reduzca la diferencia de rendimiento, ya que desarrolladores de todo el mundo colaboran para actualizar los LLM actuales.

En el contexto actual, elegir el modelo de lenguaje (LLM) de código abierto adecuado puede ser un desafío. Antes de reentrenar un LLM específico, deberíamos considerar lo siguiente:

- **Objetivo:** ¿Qué queremos lograr? Algunos LLM de código abierto están disponibles solo para investigación, por lo que debemos tener en cuenta las limitaciones de las licencias si estamos creando una empresa.
- **Necesidad:** ¿Por qué necesitamos un LLM? Si podemos alcanzar nuestros objetivos sin él, quizás no sea necesario reentrenar un modelo.
- **Precisión:** La precisión está relacionada con el tamaño del LLM. Modelos más grandes, como LLaMA o Falcon, ofrecen muy buena precisión.
- **Inversión:** Cuanto más grande sea el modelo, más recursos requerirá. Debemos considerar los costos y la infraestructura necesaria.
- **Modelo preentrenado:** Si existe un modelo preentrenado que se ajusta a nuestro caso de uso, podría ser una alternativa más eficiente.

Opciones de esta índole todavía están en fases iniciales y son pocos los que se atreven a lanzar su propio modelo. La mayoría de iniciativas están reservadas a las compañías propietarias de los LLM por su capacidad, a grandes gestores de repositorios de código (Github y GitLab) por razones comerciales obvias y a las grandes corporaciones que podrían estar repensando el ecosistema de desarrollo en el que están inmersos con soluciones in-house de vasto mantenimiento. Posiblemente pronto veamos colaboraciones entre distintos actores en este sentido.

4.2 Cambios de responsabilidades en líderes de equipos de desarrollo

Según distintas fuentes, para 2025, más de la mitad de los roles relacionados con líderes en ingeniería de software aplicarán explícitamente la supervisión por IA generativa.

Además del impacto de la IA generativa en la implementación de la tecnología, también cambia las responsabilidades gerenciales de los líderes de ingeniería del software, incluidas aquellas relacionadas con la gestión de equipos, la gestión del talento y la creación de políticas éticas. No olvidemos que actualmente, muchas compañías que prestan sus servicios de desarrollo de software o de gestión de incidencias no permiten a los miembros de sus equipos utilizar herramientas de IA generativa por temor a la falta de privacidad de los datos que introducimos en ellas o a fallas de seguridad inoculadas en el código.

Una de las principales funciones a priorizar es la gestión económica. Los costos económicos y los planes de suscripción asociados con las IA generativas señalan un alejamiento de los paradigmas económicos tradicionales. Tanto en modelos LLM de terceros como en soluciones in-house habrá que interiorizar el nuevo modelo de coste asociado a estas herramientas (pago por token, por ajuste del modelo o fine-tuning, etc). Un perfil de FinOps se antoja inevitable para estos menesteres.

Otra de las cuestiones a las que deberán enfrentarse es la discusión sobre la reducción de costes y el reemplazo de personal para centrarse en cómo estas tecnologías mejoran la productividad de los desarrolladores y mejoran la vida laboral ante los retos que se avecinan. Debido a la forma en que los modelos LLM están diseñados para generar código, no reemplazarán a los desarrolladores en el corto y mediano plazo. Si bien la IA generativa puede automatizar ciertos aspectos de la ingeniería del software, no puede todavía

replicar la creatividad, el pensamiento crítico y las habilidades de resolución de problemas que poseen los ingenieros humanos, aunque puede acompañar y asistir en muchos de esos procesos.

Lo que se esperará de los líderes en ingeniería de software es vincular eficazmente sus resultados tecnológicos con los resultados comerciales y convertirse en arquitectos del cambio.

Además deberán identificar los riesgos asociados al uso de modelos de IA generativa:

- Dependencia excesiva de la tecnología.
- Riesgos de seguridad y privacidad.
- Posibles sesgos en el código generado.

La IA generativa también puede acelerar los procesos de incorporación de nuevos miembros de los equipos. Los chatbots con tecnología de IA pueden ayudar a los nuevos empleados con preguntas frecuentes y brindarles la información necesaria para comenzar. Ello incluye la burocracia, la capacitación y otras tareas.

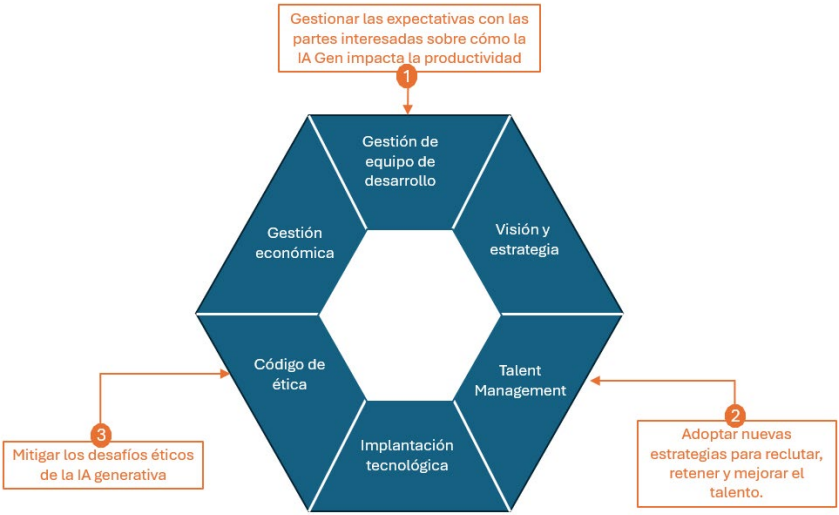
Además de las vitaminas para la contratación, **la gestión y el desarrollo de habilidades** serán el núcleo de las responsabilidades de estos líderes. Deberán trabajar con RRHH para invertir en un enfoque de habilidades dinámicas que ayuden a apoyar el talento y la gestión de los procesos de trabajo. Ello proporcionará la posibilidad de repensar roles al identificar habilidades que pueden combinarse para crear nuevos puestos.

Todos estos líderes, managers, ejecutivos, ingenieros y arquitectos de software contarán con herramientas de IA más efectivas para la toma de decisiones, como por ejemplo:

- Sin duda los chatbots de mayor relevancia como ChatGPT, Gemini, Claude, Copilot, etc, que pueden ser usados como asistentes de Inteligencia Artificial ya que pueden generar texto coherente y contextualizado relacionado con la gestión de los proyectos.

- **Jira:** Ofrece análisis predictivo para proyectos ágiles, anticipando riesgos y garantizando una entrega oportuna. Utiliza la IA para pronosticar y mitigar retrasos en los proyectos.
- **Wrike:** Proporciona información de Inteligencia Artificial sobre las cargas de trabajo del equipo, lo que ayuda a una gestión eficaz de los recursos y a la prevención del agotamiento.
- **Hummata:** Analiza la dinámica del equipo y proporciona información sobre los patrones de colaboración y las interacciones del equipo, ayudando a una colaboración eficaz y resolución proactiva de conflictos.
- **Monday.com:** Ofrece planificación de proyectos mejorada con IA, permitiendo una visión del futuro de la gestión de proyectos y facilitando la toma de decisiones informadas.

Estas herramientas de IA pueden ayudarles a tomar decisiones más informadas y basadas en datos, optimizar procesos y mejorar la eficiencia operativa.



Cómo prepararse para las responsabilidades que conlleva la IA generativa (Source: Gartner)

4.3 Retos relacionados con la privacidad y la seguridad

La Inteligencia Artificial generativa (IA generativa) ha revolucionado el desarrollo de software, permitiendo a los desarrolladores automatizar tareas y mejorar la eficiencia. Sin embargo, su uso también plantea serios riesgos de privacidad que deben ser cuidadosamente gestionados.

Exposición de Datos Personales

Las herramientas de IA generativa pueden inadvertidamente exponer datos personales sensibles durante el proceso de desarrollo o incluso código propietario que no se desea compartir con terceros. Al entrenar modelos con grandes volúmenes de datos, existe el riesgo de que información personal sea utilizada sin el consentimiento adecuado, violando leyes de protección de datos y comprometiendo la privacidad de los individuos.

Agregación y Análisis de Datos

Los sistemas de IA generativa tienen una enorme capacidad para agregar y analizar datos de múltiples fuentes. Esto puede llevar a la creación de perfiles detallados sobre individuos, aumentando el riesgo de vigilancia y discriminación. Además, la falta de transparencia en estos procesos de agregación dificulta que los usuarios sepan cómo y para qué se utilizan sus datos.

Falta de Control del Usuario

Los usuarios y desarrolladores a menudo tienen un control limitado sobre cómo se manejan sus datos en las plataformas de IA generativa. Esto puede llevar a situaciones en las que los datos se utilizan de maneras no previstas o no deseadas, comprometiendo la privacidad de los datos y exponiendo a las empresas a riesgos legales y reputacionales.

Riesgos de Seguridad en el Desarrollo de Software

El uso de IA generativa en el desarrollo de software también puede introducir vulnerabilidades de seguridad. Los modelos pueden ser manipulados para insertar código malicioso o explotar fallos de seguridad, lo que puede resultar en la filtración de datos personales y otros problemas de privacidad.

Cumplimiento Normativo

Las organizaciones deben asegurarse de que el uso de IA generativa cumple con las regulaciones de protección de datos, como el GDPR en Europa. La recopilación y el procesamiento de datos personales deben gestionarse de manera ética y legal, garantizando que se respeten los derechos de privacidad de los individuos.

En conclusión, aunque la IA generativa ofrece significativas ventajas en el desarrollo de software, es crucial que las organizaciones implementen medidas de protección de datos robustas y transparentes para mitigar los riesgos de privacidad asociados.

CAP. 5

CONCLUSIONES

Desde la mejora de la productividad en el desarrollo de software hasta la personalización de la educación y la optimización de la investigación científica, la IA generativa está demostrando ser una herramienta poderosa y versátil que está cambiando el panorama tecnológico global.

En el caso del desarrollo de software, la IA generativa tiene el potencial de transformarlo radicalmente, ofreciendo numerosos beneficios en términos de eficiencia y calidad. Sin embargo, es vital abordar los desafíos asociados para asegurar un uso efectivo y seguro de esta tecnología. A medida que la IA sigue avanzando, su impacto en el desarrollo de software será cada vez más significativo, y es esencial que desarrolladores, líderes y empresas se adapten a este nuevo paradigma para maximizar sus oportunidades.

Ya hemos visto que desde su uso más básico hasta las tendencias emergentes en IA están simplificando la creación de software, automatizando tareas tediosas y liberando a los desarrolladores para enfocarse en problemas más complejos.

La integración de herramientas de IA y sobre todo de IA generativa en los equipos de desarrollo de software supone un avance significativo en la toma de decisiones internas y la optimización de los distintos procesos dentro del ciclo de vida del software. Gracias a estas tecnologías de vanguardia, los líderes de los equipos pueden mejorar la eficiencia operativa y obtener resultados superiores en la gestión de los equipos, la gestión del talento y el reto de la privacidad y la ética.

¿Y para los no programadores? A los profanos la IA generativa también les ha proporcionado numerosas ventajas y les ha incentivado a adentrarse en el mundo de la programación sin necesidad de tener conocimientos avanzados y a capacitarse y a atreverse, por ejemplo, a automatizar tareas repetitivas con macros de Excel o a generarse plantillas con herramientas low-code amparados por chatbots de los principales LLM del ecosistema.

Ahora puede resultar más difícil saber si el código ha sido escrito por un programador experto o por uno menos experimentado con ayuda de IA ¿Esta reducción de barreras puede significar un aumento de la competencia en los equipos de desarrollo? No estrictamente. Pero sí que implica una democratización de un entorno muy tecnificado que muy pronto empezará a buscar soluciones para los usuarios pseudo desarrolladas por ellos mismos.

Si bien no hay una hoja de ruta en la adopción de la IA, aquellos que abracen estas tecnologías emergentes estarán mejor posicionados para enfrentar los desafíos del futuro y ofrecer aplicaciones de alta calidad a sus usuarios.

No hay que dudar en utilizar IA generativa para explicar, detectar y medir la deuda técnica y su impacto, aunque siguen existiendo retos relacionados con la privacidad y la seguridad. Hay compañías que prohíben a los desarrolladores utilizar chatbots debido a estos riesgos inherentes al uso de plataformas de terceros. Existe un camino por recorrer en este ámbito y los modelos LLM reentranados en nuestros servidores locales no son una mala opción aunque hay que tener en cuenta el cambio significativo en la manera de afrontar los costes.

En resumen, la IA generativa ha democratizado el acceso a la programación, permitiendo que más personas sin formación técnica se aventuren en el desarrollo de software, mejoren su productividad y adquieran nuevas habilidades tecnológicas.

Según Gartner, para el 2027 el 70% de desarrolladores profesionales usarán herramientas 'aumentadas' con IA. Estamos en una primera etapa de impulso y la madurez de la adopción y el uso de 'herramientas Gen AI' aún sigue siendo comprensiblemente baja, pero anuncia una carrera meteórica.

Todos los actores relacionados con el ciclo del desarrollo de software se encontrarán en una desventaja significativa si no reconocen y se adaptan a estos cambios, enfrentando el riesgo de ser reemplazados por aquellos que adoptan esta tecnología disruptiva.

Esta es realmente una herramienta útil aunque, por otro lado, puede llegar a reescribir el futuro del gremio. Y aunque nadie puede adivinar qué pasará en el porvenir, a día de hoy podemos contestar que se trata de un gran aliado que vale la pena utilizar. Ante la distopía del 'mito de la muerte del programador por la IA', la verdad es que morir, no va a morir, pero transformarse seguro que sí.

5.1 Bonus Track: Herramientas de IA para el desarrollo de software

He aquí unas cuantas de las muchas herramientas que ya están empezando a formar parte del ecosistema del desarrollo de software.

Codesnippet

Codesnippet es una excelente herramienta para desarrolladores impulsada por ChatGPT. Utiliza el aprendizaje automático y el procesamiento del lenguaje natural para generar automáticamente fragmentos de código a partir de descripciones en lenguaje natural. Así, el flujo de trabajo del desarrollador se agiliza y las tareas de programación repetitivas se automatizan.

SpellBox

Esta herramienta genera los fragmentos de código necesarios basándose en sencillas indicaciones. Por lo tanto, puede abordar incluso los retos de programación más complejos en cuestión de segundos. En general, es una excelente herramienta asistente de codificación para aficionados, profesores y codificadores profesionales.

GitHub Copilot

GitHub Copilot es una revolucionaria herramienta de AI asistente de codificación que tiene el potencial de cambiar por completo la forma en que los programadores escriben código. La plataforma utiliza actualmente el códice Open AI y sugiere códigos y funciones en tiempo real desde el editor.

AiXcoder

Puede producir fácilmente código a nivel de método traduciendo el lenguaje natural a código mediante AiXcoder. Además, esta herramienta de IA proporciona completado inteligente de código para líneas enteras de código o grupos de líneas.

Codex

El modelo Codex destaca en Python y también tiene un dominio notable de otros lenguajes. Por ejemplo, JavaScript, Go, Perl, PHP, Ruby, Swift, TypeScript, SQL e incluso Shell. Tiene una amplia gama de aplicaciones, desde el autocompletado de código hasta la producción de sofisticados fragmentos de código.

Codeium

Codeium goza de una ingente popularidad y es usado por miles de desarrolladores para autocompletar, chatear con AI y buscar código más rápido. También está disponible para equipos, lo que permite a empresas de todos los tamaños colaborar codificando juntas.

OpenAI Codex

Está disponible como API. Sería adecuado para los desarrolladores senior que quieren aprovechar la finalización de código a través de una API. O, si lo que busca es crear herramientas de codificación de IA, puede aprovechar la API de OpenAI.

Google Codey

La API Codey de Google forma parte del conjunto de API PaLM que ofrece la generación de código a través de API. Code API es compatible con lenguajes de programación populares como Java, TS, JS, Go, Python, etc.

AlphaCode

AlphaCode de DeepMind es un sistema de IA que construye códigos en respuesta a explicaciones en lenguaje natural de un problema utilizando modelos de lenguaje basados en transformadores los modelos de redes neuronales en los que se basan los LLM). Puede resolver problemas complejos de programación que implican pensamiento crítico, lógica, algoritmos, codificación y comprensión del lenguaje natural.

Tabnine

Tabnine es un potente asistente de IA que ayuda a los desarrolladores a potenciar su creatividad mientras codifican. Con sus funciones avanzadas y su interfaz fácil de usar, ofrece una forma más inteligente de codificar, lo que lo convierte en la selección ideal para millones de desarrolladores de todo el mundo.

AskCodi

Askcodi es un asistente de código de IA desarrollado a través de Assistiv. ai, que ayuda a los desarrolladores a codificar de forma más rápida e inteligente. Está conectado con los editores más comunes, incluyendo Visual Studio Code, Sublime Text, Atom, y más.

Además, es compatible con una amplia variedad de lenguajes y marcos de programación, incluyendo Python, Java, JavaScript, C#, Ruby, PHP, CSS, React, Angular, Vue, y más. Por lo tanto, puede evitar errores, aprender nuevas habilidades, y mejorar su productividad como codificador.

IntelliCode

Intellicode de Visual Studio es un motor de codificación impulsado por IA. Utiliza el aprendizaje automático para ofrecer ideas y recomendaciones inteligentes a los desarrolladores. Además, agiliza el proceso de desarrollo impulsando la productividad y ofreciendo recomendaciones.

Replit

Replit es un IDE potente y versátil que le permite desarrollar software con el poder de la IA. Hace que la codificación sea flexible, más eficiente y más colaborativa. Además, puede incluso codificar directamente utilizando su navegador web. Como resultado, sus habilidades de codificación ir al siguiente nivel con la ayuda de esta plataforma, si usted es un principiante o un desarrollador experimentado.

Blackbox

Blackbox es una sofisticada plataforma que permite a los desarrolladores crear, desplegar y gestionar fácilmente modelos de aprendizaje automático. Fue diseñada para ayudar a los programadores a escribir código más rápido que nunca. La plataforma es compatible con 20 lenguajes de programación y crea asombrosos fragmentos de código.

AutoRegex

AutoRegex es una potente aplicación impulsada por IA que utiliza el Procesamiento del Lenguaje Natural (NLP) para simplificar el desarrollo de Expresiones Regulares (RegEx). AI traducir el inglés a RegEx, le permite generar de forma sencilla patrones RegEx sin necesidad de grandes conocimientos de sintaxis.

Amazon CodeWhisperer

Amazon CodeWhisperer utiliza miles de millones de líneas de código de proyectos del mundo real para proporcionar sugerencias instantáneas de última generación, con todo tipo de funciones, desde fragmentos hasta funciones completas, incluso para API desconocidas.

Kodezi

Kodezi ofrece una solución integral de codificación para desarrolladores, estudiantes y equipos. Esta suite de productividad avanzada automatiza diversas actividades de codificación, como la optimización, los comentarios, los docstrings, el resumen, la depuración y la generación de código. Los desarrolladores disponen de una plataforma conversacional profesional. Dentro de esta plataforma, pueden realizar consultas, crear código y explorar sus recursos de codificación. Es una solución elaborada con tecnología punta que no sólo resuelve los errores, sino que también ofrece explicaciones exhaustivas sobre su aparición y valiosas sugerencias para prevenir futuros fallos.

What The Diff

El servicio What The Diff mejora la revisión del código mejorando la calidad de las descripciones y revisiones de las pull requests. Con la ayuda de la IA, acelera el proceso de fusión de pull requests de forma eficaz sin almacenar el código de ningún usuario. Proporciona explicaciones contextuales para los cambios de código, dando a los usuarios el control sobre los repositorios, el análisis, la inclusión/exclusión de archivos y la generación de comentarios.

DocuWriter.ai y Scribe

Generan automáticamente documentación clara y precisa a partir de la base de código, ahorrando tiempo y reduciendo las inconsistencias.

CAP. 6

BIBLIOGRAFÍA

- Inteligencia artificial en desarrollo de software: Tendencias emergentes y futuro | Openwebinars - Marzo 2024
- Las 10 mejores herramientas de IA para desarrolladores en 2024 | ScreenApp Blog - Marzo 2024
- Generative AI changes software engineering leaders responsibilities - Gartner - Agosto 2023
- How to use generative AI to boost developer productivity - Gartner - Diciembre 2023
- Wikipedia - Lenguajes de programación - Junio 2024
- AI and Machine Learning For Coders: A Programmer's Guide to Artificial Intelligence - Lawrence Moroney - Octubre 2020
- The Art of Functional Programming - Minh Quang Tran - Octubre 2022
- Qué es Copilot de GitHub y cómo funciona esta inteligencia artificial que te ayuda a programar | Xataka Basics - Febrero 2023
- ChatGPT Para Desarrolladores: Casos Clave En El Desarrollo De Software | ApiumHub - Junio 2023
- Transformación del desarrollo de software: El impacto de Modelos Grande de Lenguaje (LLM) - Analytics Lane - Diciembre 2023
- Code Llama, a state-of-the-art large language model for coding | Meta - Actual
- Meta lanza Code LLaMA 2, una IA 'libre' especializada en generar código fuente: su rendimiento es mayor que el de GPT-4 | Genbeta - Enero 2024

- The Future Of AI In Software Development - FortySeven - Enero 2024
- AI and the Future of Software Development - InfoWorld - Agosto 2023
- How Will AI Affect the Future of Software Development? - Railwaymen - Mayo 2024
- The Future of Programming: How AI is Shaping the Industry - Medium - Febrero 2024
- Is There a Future for Software Engineers? The Impact of AI - BrainHub - Junio 2024
- Will AI Replace Software Engineers? Exploring the Future of Software Development - Baylor University - Noviembre 2023

INESDI + BUSINESS TECHSCHOOL

www.inesdi.com

RED DE EDUCACIÓN SUPERIOR

Planeta Formación y Universidades