Tabla de Contenidos

Entrevista en Diferido: Joaquín Herrero

Tabla de Contenidos

Reproducimos aquí la entrevista que hizo José Jiménez a Joaquín Herrero en su canal de Telegram en mayo de 2020. Enlace a la entrevista original: https://t.me/entrevistaendiferido/1193

Empezamos una nueva a entrevista a un programador cuyo nombre es Joaquín Herrero, también uno de los integrantes del podcast sobre Filosofía, La filosofía no sirve para nada (https://filosofias.es/wiki/doku.php/podcast/start). Pero lo mejor es que se presente el entrevistado.

¿Te podrías presentar en unas líneas?

Hola, gracias por traerme a tu canal. Me llamo Joaquín Herrero Pintado y trabajo como responsable de los servicios de teletrabajo y colaboración en línea en una administración pública. Las tecnologías que administro allí son bastante variadas: Xen, Netscaler, Windows Server 2016, Ubuntu 18.04, Docker, Citrix, Owncloud, OpenBSD y Dokuwiki.

Por otra parte también soy graduado en Filosofía y desarrollo una investigación desde hace ya años sobre temas que están bastante relacionados con la computación entendida de forma conceptual: complejidad y la realidad como información.

También a lo largo de estos años he impartido cursos y talleres tanto sobre aspectos técnicos informáticos como sobre la reflexión filosófica sobre las tecnologías de la información.

Suelo tener siempre algún que otro *side project* que suele consistir en programación de algo que me resulte interesante y estos últimos años he programado varios chatbots para Telegram en lenguaje Go, uno de los cuales se está usando en un ensayo clínico para verificar la efectividad de los chatbots para complementar la atención médica presencial.

También, como comentas, hace poco más de un año comencé el podcast “La filosofía no sirve para nada” en el que con un grupo de personas muy interesantes e interdisciplinares echamos una mirada

filosófica al presente y al cruce entre Ciencia, Tecnología y Sociedad (CTS), que es un campo que no mucha gente conoce pero que se practica en la filosofía desde los años 60 del siglo XX.

Para empezar , unas cuantas preguntas cortas para conocerte mejor a tí y tu entorno tecnológico.

¿Qué ordenador utilizas habitualmente?

Mi máquina desktop principal para casi todo lo que no es trabajo oficial es un ThinkPad x220 con 8GB de RAM. El form factor de esta máquina es sencillamente perfecto y su teclado me parece insuperable.

Mi desktop para temas de trabajo es un portátil ACER V5-571G (basado en Core i5) con 8 GB RAM y monitor externo HP 24er.

Mi máquina favorita de movilidad cuando no llevo el x220 suele ser una tablet Teclast X98 Plus II con el Universal Mobile Keyboard de Microsoft, router 4G/Wifi y ratón Bluetooth de Microsoft. Es decir, teclado pequeño (como a mí me gustan) y resolución de pantalla grande en poco espacio: 2048×1536 en 10 pulgadas. Esta máquina me hace muy feliz a pesar de que en los foros especializados suele ser muy criticada porque la antena WiFi es muy deficiente y no alcanza a ver puntos de acceso que estén a varios metros de distancia (esto es verdad verdadera) y porque su Android se ha quedado obsoleto (también verdad). Pero como esos dos problemas no me afectan (eliminé Android y el router WiFi está a 10 centímetros de la máquina) pues solo me da satisfacciones.

¿Sistema operativo?

En el x220: OpenBSD 6.7

En el ACER: Windows 10 (1909)

En la Teclast: eliminé Android (traía la versión 5 y no se podía actualizar). Jugando con las particiones UEFI le di los 64 GB de espacio a Windows 10 (1909) y lo tengo completamente actualizado.

¿IDE o editor para programar?

Es una pregunta corta pero me resulta difícil dar una respuesta corta. Lo intentaré. Últimamente se entiende que un IDE es una pieza de software que te instalas y que contiene todo tu “entorno”. Este es el caso de VSCode. Digo últimamente porque yo procedo de una cultura informática en la que el entorno de desarrollo era el propio sistema operativo, algo que incluso ahora tiene su propio nombre: UNIX As IDE. Desde ese punto de vista no uso IDE, uso editores. Pero como mi visión de qué es un IDE no coincide con esa pues mi respuesta sería que sí, sí uso IDE, pero no uso un “programa como IDE”, uso todo el sistema UNIX como IDE.

Mis editores básicos son dos, según el trabajo a realizar o igual según mi humor de ese día: vim(1) y ed(1). Las razones son históricas, por tanto muy personales y difícilmente reutilizables por otras personas, pero os las cuento, como curiosidad, a continuación.

Mis primeros pasos como programador fueron con un editor de línea de los mainframe de Univac llamado @ED que era descendiente directo del editor usado en el “Project MAC” del MIT (años 60) pero con un lenguaje de programación integrado.

Cuando, como fue mi caso, lo primero que conoces como editor es algo como @ED, un editor con un lenguaje de programación completo en su interior, mantienes en la mente grabado a fuego para toda tu vida la idea de que editar un archivo es hacer programas que lo cambien y no cambiarlo tú a mano moviéndote fácilmente por el texto.

Usando el lenguaje de programación del editor @ED llegué a hacer un generador de programas COBOL que me generó más de 1000 programas diferentes de consulta y creación de ficheros y bases de datos a partir de un 'esqueleto tipo' tan solo respondiendo a un cuestionario de preguntas: un editor de línea usado como lenguaje de programación para automatizar cambios. Su belleza no está en su interfaz sino en su potencia. Si queréis saber algo más sobre @ED tuve la fortuna de poder entrevistar por email a uno de sus creadores, Jerry Saltzer, que me contó muchas cosas que no están publicadas en ningún sitio y algún que otro cotilleo que me hizo prometer no publicar mientras él viviera. Aún vive. La entrevista aquí:
https://misdocumentos.net/wiki/doku.php/mainframes/univac/ed/historia

Editores como ed(1) o @ED no te facilitan moverte por el texto (no es su objetivo) pero te facilitan automatizar los cambios, esa es la grandeza de los editores de línea, que son auténticos lenguajes de programación.

Pero hay que reconocer que requieren un mindset diferente a un editor convencional. Tengo que decir que ed(1) es una versión muy reducida de @ED y sin su potente lenguaje de programación, pero aún así es muy potente por su facilidad para integrarse con el resto de utilidades del sistema y por eso me sirve para rescatar ese marco mental que cité antes: editar es programar. Por eso vuelvo a ed(1) siempre que puedo para disfrutar de editar de otra forma y, de paso, mantener afiladas mis habilidades con cosas como find(1) y awk(1) para cuando surjan problemas en el sistema y haya que hacer scripts de mantenimiento.

Vim(1) es mi alternativa a ed(1) para disfrutar de las ventajas de sus plugins, un ecosistema maravilloso. También tengo grabados a fuego los toques de teclado que necesito hacer en Vim y me siento como en casa usándolo. Sé que hay una guerra entre programadores que usan Vim y los que usan Emacs y me hace mucha gracia que la haya. Pero esa guerra debe esconder algo profundo que no he llegado a averiguar: he sido incapaz de aprender Emacs y, sin embargo, con Vim me sentí cómodo desde el primer momento. Igual hay que pensar la razón de esto filosóficamente.

Finalizando. Yo solo le veo ventajas a usar el sistema operativo (shell y utilidades) como IDE: mantiene a punto mis habilidades para hacer scripting y solucionar cualquier problema del sistema. Pero mi punto de vista no es algo que suela recomendar. No evangelizo sobre editores pero tengo convicciones muy firmes sobre ellos. Si alguien que empieza a programar me preguntara qué IDE usar le recomendaría que usara VSCode: es la cultura actual y no puedes sustraerte a ella. Pero también le diría que recordara que fuera de un “software como IDE”, fuera de VSCode, hay un universo de herramientas que no han cambiado en décadas porque son perfectas como caja de herramientas para solucionar problemas de edición o problemas de mantenimiento de sistemas.

Lo siento, no conseguí ser breve. Esta pregunta ha tocado uno de mis temas favoritos.

¿Herramienta que siempre está en tu equipos?

En una máquina Windows no puede faltar el MobaXterm para conectar por terminal con mis máquinas de backend. También instalo PuTTY/Kitty por si acaso, aunque acabo usando Moba siempre.

En una máquina Android no me puede faltar Termux por la misma razón: uso las máquinas Desktop pero donde realmente vivo es en el backend.

Y siempre instalo en toda máquina Desktop un procesador de textos que me permita escribir en Markdown con comodidad. Desde que descubrí Ghostwriter es mi procesador de textos favorito para redactar artículos. Lo tengo instalado en todas las máquinas OpenBSD con entorno gráfico y en todos los Windows que uso.

Y en cualquier OpenBSD con entorno gráfico siempre hago el ritual de instalar DOSBOX y la versión para DOS de @ED. Sí, sigue existiendo y es igual de briullante manejando programas en Go.

¿Un libro de filosofía que un programador debería leer?

Esta es fácil, hay muchos, pero uno que se lee bien si tienes mente filosófica es Philosophy and Computer Science, de Timothy R. Colburn, que es filósofo e informático (qué poco me gusta la palabra “informático”)

Lo recomiendo entero, de portada a contraportada, pero especialmente meterse en vena el capítulo 12: “Software, Abstraction and Ontology”

> Como has comentado tienes un podcast sobre filosofía con un grupo de persona, quería centrarme en el termino de filosofía. Mucha gente conoce el termino y es utilizado en diversos ámbitos como el deporte (filosofía del futbol) o en desarrollo de software(filosofía del software libre). Pero realmente poca gente la estudia o simplemente no sabe para que sirve.

¿Para qué sirve la filosofía en la "vida real"?

El tuit que tengo fijado en Twitter desde hace mucho tiempo creo que responde a esa pregunta: “Philosophy moves in conceptual spaces where philosophers produce frameworks for thinking about a given problem.”. La filosofía produce “frameworks”, que es una palabra que muchos programadores conocen bien. En programación un framework es un entorno de herramientas que te permite abordar un problema de forma más sencilla que si no lo tuvieras. En filosofía un framework sería un “marco de pensamiento”, abstracciones y conceptos que sirven para ver más claro cómo encarar un problema. Pues el caso es que la filosofía construye esos frameworks y luego los usa, aunque esos mismos frameworks pueden ser usados por cualquiera, por alguien sin conocimientos filosóficos. Pero al usarlos estás filosofando sin darte cuenta. Sobre esta capacidad de la filosofía es de obligada lectura

el libro de Gilles Deleuze titulado “¿Qué es la filosofía?” donde explica muy bien cómo la filosofía construye conceptos y cómo son usados después por las ciencias.

Típicamente los framework filosóficos permiten hacer una cosa muy concreta: la evaluación crítica de creencias. También lo llamamos “pensamiento crítico”. Es decir, la filosofía sirve para problematizar las cosas que damos por sentadas, pensar “out of the box” para llegar a ver problemas nuevos o nuevas formas de pensar los problemas que existen. Pongo ejemplos para que se entienda.

La forma novedosa de reflexionar sobre la naturaleza que surge en Grecia (llamada “filosofía natural”) y la posibilidad de matematizar el funcionamiento de la naturaleza (algo que hicieron filósofos medievales, siglos XII y XIII, en Oxford) hicieron posible que en el siglo XVII surgiera una obra como la de Isaac Newton, cuyo título, quiero recordar, es “Principios matemáticos de filosofía natural” aludiendo, por tanto, a esas dos innovaciones filosóficas en las que se apoya su teoría sobre la gravedad (con la que podemos usar naves espaciales, por ejemplo). Galileo y Newton dan comienzo a la ciencia tal como la entendemos ahora, con una parte experimental y una reflexión teórica. La parte de reflexión teórica es la aportación de la filosofía a nuestro concepto de ciencia. ¿Para qué ha servido entonces la filosofía antigua de Grecia? Una respuesta bastante acertada sería: para construir nuestro concepto de Ciencia.

No podemos decir algo concreto que surgirá al pensar filosóficamente. En este sentido no sirve para nada concreto. Pero sirve para pensar distinto, para problematizar lo que damos por sentado. ¿Dónde te llevará esto? Pues a cada uno lo llevará a un lugar diferente. Pero seguro que será un lugar interesante.

En matemáticas e informática también nos ha pasado. Hay muchas verdades establecidas en la informática o en la matemática que tienen su origen en investigaciones filosóficas. La lógica fue tratada originalmente por Aristóteles y es el fundamento de la informática tal como la entendemos ahora. El filósofo Leibniz propuso el cálculo binario para resolver problemas y ahora todos los problemas se resuelven mediante cálculos que en última instancia son binarios. El desarrollo del cálculo integral en matemáticas se hizo a la vez por Newton y Leibniz porque ambos partían de supuestos filosóficos diferentes, era un problema filosófico en origen pero ahora es una cosa muy práctica, pues con una integral podemos medir el área de superficies irregulares. Esto es posible porque ahora consideramos normal hablar de “infinitesimales”, pero ese concepto hubo que construirlo y se construyó filosóficamente y con muchos debates. Hay un libro de Amir Alexander titulado "Infinitesimal" que cuenta esta historia muy bien. Matemáticas y filosofía han estado íntimamente relacionadas durante toda la historia.

La Inteligencia Artificial es seguramente el tema actual donde más conexión hay entre filosofía y computación. Hemos tocado este tema varias veces en el podcast.

Es decir, filosofando acabas siempre en la “vida real” pero nunca sabes dónde exactamente. Eso sí, cuando llegas a la “vida real” después de haber pensado filosóficamente, es bastante probable que ese aspecto de la vida real ya no lo veas como antes, ahora puedes ver formas nuevas de pensar esa realidad. Has creado un “framework” para ver esa realidad de una nueva manera. Y ese framework lo usarán filósofos para seguir pensando, también lo usarán científicos para construir explicaciones sobre el mundo y también lo usarán ingenieros para construir artefactos.

¿Qué aporta la filosofía a un programador?

Si como programador tu interés es ser eficaz al resolver los problemas que te encargan y resolverlos

de forma eficiente pues no sé si la filosofía aporta algo. Ese programador que busca la eficacia y la eficiencia lo que quiere es transitar rápidamente una carretera que ya está construida. Muy razonable. Yo soy ese programador en muchas ocasiones.

Pero a veces quiero ser otro tipo de programador. Lo explico. Si como programador estás insatisfecho por la forma en la que habitualmente se suele resolver algo, es decir, si estás insatisfecho con la carretera que tienes que recorrer, entonces la forma de pensar filosófica puede ayudar, porque lo que quieres es descubrir un nuevo "framework" para resolver el problema. Por ejemplo, un nuevo algoritmo. O un nuevo paradigma de programación. Si miras en Wikipedia verás que "paradigma de programación" se define así: "Un paradigma de programación es una propuesta tecnológica adoptada por una comunidad de programadores. Representa un enfoque particular o filosofía para diseñar soluciones". Un programador está en muy buena posición para cuestionar el mismo paradigma de programación que suele usar, precisamente porque lo conoce muy bien.

Si como programador quieres experimentar nuevas formas de programar o crear nuevos algoritmos, entonces estás usando la forma filosófica de pensar y aplicándola a tu campo concreto. Ahora tenemos diversos paradigmas de programación: la programación funcional y la declarativa, por ejemplo. Durante mucho tiempo hemos tenido bastante arrinconada la programación funcional y hemos explorado la declarativa, creando cosas tan interesantes como la programación orientada a objetos. Todo esto ha sido inventado por alguien. Estos modelos de programación no nos han caído del cielo en un meteorito. Alguien pensó filosóficamente y cuestionó el framework habitual. La instrucción "Go To" ha sido objeto de debates como lo fueron en su día los números infinitesimales, y tienes formas de programar "sin Go To" y "con Go To". El concepto de "clases" propio de la programación orientada a objetos ha sido cuestionado como un framework defectuoso, una forma mala de pensar. De hecho incluso James Gosling, padre del lenguaje Java, dijo que lo que quitaría de Java son las clases, precisamente. Y la forma de pensar alternativa es la "herencia de interfaces". Esa es la razón por la que yo aprendí el lenguaje de programación Go. Quería experimentar ese nuevo framework. Elegir un lenguaje de programación y no otro puede ser ya en sí mismo una decisión filosófica.

Programar puede ser un acto de ingeniería, una forma eficaz y eficiente de solucionar problemas, pero también puede ser una actividad filosófica, de exploración y cuestionamiento sobre la forma como resolvemos problemas.

La siguiente es una pregunta fija en todas las entrevistas ,solo cambia el contexto en función del entrevistado. En tu caso, tenía claro el contexto pero no muy bien la pregunta, pero creo que sería interesante preguntarte...

¿Qué añadirías a la Filosofía?

Una "s".

Mi proyecto filosófico en Internet se llama "filosofias.es" precisamente porque nombrarla en singular hace que la gente pierda de vista que la filosofía es un territorio muy complejo, es una red de conocimiento con muchos rincones, algunos de los cuales son muy conocidos y explorados en la actualidad (ética, política) y otros son mucho menos conocidos.

Sin entrar a fondo en este tema la idea es que en la Academia se enseñan unos conocimientos concretos por cuestiones históricas, y habitualmente se descarta, por ejemplo la filosofía de la matemática, la filosofía de la información o la filosofía de la física que pasan a ser filosofías poco conocidas por el público general.

Esto solo consigue que la gente tenga unos prejuicios sobre la filosofía que asombran a cualquiera que la conozca desde dentro.

- ¡Ah!, ¿la filosofía también trata sobre X?
- Por supuesto, verás.. (aquí seguiría una charla de varios minutos)

Sustituye X en el diálogo anterior por cualquier tema y el diálogo sigue valiendo.

¿Qué eliminarías de la Filosofía?

La mayúscula.

Solemos escribir “Filosofía”, así en mayúsculas, cuando nos referimos al cuerpo de conocimientos que se transmiten académicamente. Y el resto son “filosofías”, es decir, formas de hacer las cosas, opiniones sobre los temas, puntos de vista más o menos separados de la norma.

No me gusta esa distinción porque hay mucha “filosofía” que no suele enseñarse en el Grado y no se considera “Filosofía” con mayúsculas. Y hay muchas ideas mayúsculas en esas filosofías poco conocidas.

También eliminaría la compartimentación.

La filosofía engloba todas las formas en las que los seres humanos nos acercamos a la naturaleza, a otros seres humanos, a los animales o a los objetos. Y por eso no se puede desgajar ningún tipo de conocimiento de la reflexión filosófica. En el podcast hemos tocado tanto las ideas matemáticas de Gödel, las ideas del transhumanismo, las ideas religiosas, la naturaleza del tiempo, la inteligencia artificial, la creatividad, las emociones o las teorías sobre la realidad como simulación. Y para eso no nos tenemos que salir de la filosofía en ningún momento. La filosofía te enseña a construir marcos de pensamiento y todos los temas que acabo de indicar necesitan marcos de pensamiento para poder pensar los problemas que se originan dentro de cada uno de esos temas.

Eliminaría, por tanto, el pedestal en el que está subida cierta filosofía y la reducción de la filosofía a ciertos temas.

¿Qué modificarías de la Filosofía?

Su identificación con las humanidades y solo con las humanidades.

La filosofía no solo trata de lo humano, también trata sobre la naturaleza, sobre lo que hay “ahí afuera”, en la realidad. Así comenzó la filosofía, como una física. Y sigue siéndolo, más aún ahora que la física propone cada vez con más frecuencia ideas sobre la naturaleza de las que no cabe experimento posible de momento. En este punto la física y la filosofía comparten un territorio muy interesante: el pensamiento especulativo.

Una de las ideas más fascinantes sobre esto es la idea de que “la realidad consiste en información”. Esto a un informático debería de parecerle interesante. Pero no habrá muchos informáticos aún que sepan que hay muchos filósofos investigando por ese lado.

Yo haría que el libro de Víctor Gómez Pin titulado Tras la Física. Arranque Jónico y Renacer Cuántico de la Filosofía fuera libro de texto de una asignatura de grado. Por lo menos en la UNED se siguen manteniendo, que yo sepa, tres asignaturas de Historia de la Ciencia en las que se estudia el impresionante libro Historia de la Ciencia de Carlos Solís y Manuel Sellés, que llega hasta la ciencia contemporánea. Yo he tenido que explicar el modelo estándar de la física en un examen de filosofía. Eso está muy bien, ojalá la UNED siga manteniendo esa asignatura (y el alto nivel de exigencia de la misma) pero no es suficiente.
A pesar de la intensa conexión de la filosofía con la física, la informática y la matemática se sigue considerando que la filosofía son “humanidades” e incluso se identifica con la psicología y los libros de autoayuda. Una aberración, pero es lo que hay. Desde nuestro podcast tratamos de explorar todo el territorio que abarca la filosofía, tanto el territorio humanístico como el científico, el tecnológico, el cosmológico o el matemático. Porque todos esos son territorios igual de naturales para la reflexión filosófica.

No solo es la ciencia el territorio de la filosofía que la gente suele desconocer. ¿Sabías, por ejemplo, que hay tal cosa como la “filosofía de la danza” o la “filosofía del cuerpo”? Ya hemos explorado estos temas en nuestro podcast de la mano de una filósofa autora de un libro sobre filosofía y danza. Y estamos preparando episodios que hacen una reflexión filosófica sobre el cuerpo, un tema que ha sido poco tratado por la filosofía y en el presente se vuelve más necesario si cabe.

Sí, se puede hacer una reflexión crítica filosófica sobre la propia filosofía y cuestionar su adecuación a los tiempos. La filosofía, decía Hegel, es pensar el presente. Cualquier cosa del presente es competencia de la filosofía.

Cambiemos de tema, quiero preguntarte por la Inteligencia Artificial , un campo que avanza muy rápido y obtiene muchos progresos en diferentes ámbitos. Pero si analizamos detalladamente, vemos que esa IA depende mucho de la capacidad de procesamiento y los datos que le proporcionan, en ambos casos de gran cantidad. Si lo comparamos con la Inteligencia Humana, ha sido capaz de crear grandes cosas con un hardware mas limitado (cerebro) y con muchos menos datos, incluso es capaz de crear otro tipo de inteligencias (IA).

¿Es posible una IA que sea mas inteligente que la Inteligencia Humana? En caso afirmativo ¿Que requisitos se deben cumplir?

Bueno, para poder responder a esto tendríamos que tener una definición de en qué consiste la inteligencia y, una vez que sepamos qué es, tendríamos que disponer de una buena forma de medirla. Como no está nada claro que tengamos ni lo uno ni lo otro, te comentaré qué piensan sobre

esto algunos autores con los que estoy de acuerdo.

Las respuestas más interesantes a esta cuestión creo que las dan Antonio Damasio, Judea Pearl y Gary Marcus. Las resumo a continuación.

En octubre de 2019 resumimos las ideas de Damasio y Pearl en este hilo de Twitter:

“Qué añadir a un robot para que surja la inteligencia creativa: deben gestionar su propia supervivencia, sentir dolor/placer/empatía, comprender la causalidad y ser afectados por contrafácticos. Un robot así nos permitiría estudiar filosóficamente la consciencia. El trabajo se centra en qué materiales permitirían a un robot ser vulnerable y sentir dolor pues se propone la necesidad de supervivencia como una clave de la creatividad. Pero nos han llamado la atención otras cosas filosóficamente relevantes. Por ejemplo, el hecho de que haya que implementar en el robot la comprensión de qué es causalidad y qué es un contrafáctico. El aprendizaje no consistiría solo en buscar correlaciones en muchos datos pues hay casos individuales tan relevantes que deben cambiar el comportamiento. La 'inferencia causal', propuesta por Judea Pearl y bien resumida en su libro 'The Book of Why', planteaba la necesidad de incluir razonamiento contrafáctico en el campo de la Inteligencia Artificial. Parece que su mensaje está calando, como él mismo dice.”

Estuvimos hablando ampliamente sobre esto en el episodio 17 del podcast.

Estas propuestas van en la línea de las críticas que está haciendo últimamente Gary Marcus. Él no está de acuerdo en denominar “inteligencia” a lo que se hace con las redes neuronales en la actualidad. Hemos abandonado, dice él, la inteligencia artificial simbólica y nos hemos entregado a buscar patrones y correlaciones en océanos de datos. Marcus propone mezclar ambos enfoques.

Estuvimos hablando sobre esto en el episodio 15 del podcast.

Me interesan especialmente las propuestas de Judea Pearl y su idea de “contrafácticos”, es decir, que las IAs puedan aprender a partir de sus propias acciones en el mundo y no a partir de nuestras acciones recogidas en datos.

Los contrafácticos no necesitan Big Data para funcionar con inteligencia y, de hecho, nuestra inteligencia tampoco parece que se base en un análisis exhaustivo de toda la información disponible. Parece, más bien, que funcionamos con heurísticas o atajos cognitivos y buscando las causas de lo que observamos.

El trabajo de Pearl consiste en buscar los algoritmos que nos permitan describir cómo hacemos nosotros para encontrar causas y poder hacer máquinas que hagan lo mismo. A esto dediqué mi Trabajo Fin de Grado de filosofía titulado “Causalidad en escenarios de complejidad” donde comparé las estrategias de búsqueda de correlaciones en datos (redes neuronales) con las estrategias de búsquedas de relaciones causales entre eventos (los contrafácticos que propone Pearl).

Cuando estudiaba en la universidad Informática de sistemas, impartían mas asignaturas relacionadas con las matemáticas(Cálculo,Álgebra, Matemáticas Discretas y otras) que asignaturas de programación, si preguntabas porque esa diferencia te respondían que el conocimiento de las matemáticas te sirven de base para entender los conceptos de programación, así como es una buena forma de aprender resolver problemas complejos.

Cuando acabas los estudios y empezabas tu vida laboral aprendías muchas conceptos, lenguajes, librerías o framework, pero no encontrabas una relación muy directa con las matemáticas. Es verdad, que en algunos ámbitos estaban muy presentes. Pero, en comparación con la programación en general, estaba restringida a unas áreas muy especificas aunque importantes. Mi duda siempre ha sido: ¿para ser un buen programador necesitas conocimientos en matemáticas?

¿Para ser un buen programador necesitas conocimientos en matemáticas?

Pues eso que dices es cierto y al mismo tiempo tiene sentido y es un sinsentido. Me explico. Hay varios tipos de programador muy diferentes entre sí. Por ejemplo, no necesita las mismas técnicas de programación un programador de aplicaciones de negocio que usa PHP que el programador que crea el lenguaje PHP y programa compiladores. Tampoco necesitan las mismas habilidades el programador que crea un algoritmo de compresión que el programador que incluye ese algoritmo en un script de bash. Hay un tipo de programador (que no es la mayoría) que necesita ciertas matemáticas para programar y otro tipo de programador más habitual, que no las necesita. Sin embargo todos ellos necesitan (o sería conveniente que tuvieran) algo que se llama “mathematical thinking”. Sobre esto último recomiendo echar un vistazo el libro “Introduction to mathematical thinking” de Keith Devlin.

Como te dije yo soy funcionario de la administración general. En mi primera oposición (para programador) la mitad del temario era sobre matemáticas: ecuaciones lineales y no lineales, cálculo matricial, probabilidad, estadística, sistemas de numeración, combinatoria, progresiones, cálculo de interés simple y compuesto. Sin embargo en la última oposición que he hecho (gestión informática) no había nada de matemáticas y mucha metodología y conceptos abstractos de programación. Creo que en la época de mi primera oposición se entendía la informática como construcción de sistemas y desarrollo de algoritmos y según hemos ido hacia mayores niveles de abstracción ahora mucha informática consiste en una actividad de muy alto nivel que consiste en aplicar metodologías, integrar librerías y programar llamadas a dichas librerías de forma que se resuelva el problema desde un punto mucho más abstracto que antes.

En el libro Coders at Work (que recomiendo para cualquier programador al que le interese la reflexión crítica sobre su trabajo) Peter Seibel entrevista a 15 programadores muy relevantes y a algunos de ellos les hace la misma pregunta que me acabas de hacer: ¿es necesario saber matemáticas para programar?.

El libro deja claro que esta cuestión surge porque Dijkstra dijo en su famoso trabajo On the cruelty of really teaching computing science que la programación es una rama de la matemática aplicada. ¡Boom! Programar es hacer matemáticas. Y Donald Knuth en su archiconocido libro The Art of Computer Programming puso tanta matemática que es dificilísimo poder leerlo si no sabes matemáticas. Es decir, que parece que hemos heredado esta idea de la importancia de la matemática en la programación debido a dos de los padres de la programación actual: Dijkstra y Knuth. ¿Será que en sus tiempos era más importante y ahora no? Puede ser, pero eso es lo que hay que ver entonces.

Siebel en “Coders at Work” entrevista a Douglas Crockford, creador del lenguaje Javascript, y le hace

esa pregunta. Crockford responde diciendo que las matemáticas son importantes a la hora de programar pero que también lo es saber escribir con fluidez porque un programador tiene que escribir documentación y especificaciones y si alguien no es capaz de usar el lenguaje con precisión tendría muchas dificultas den su trabajo. No puedo estar más de acuerdo con él.

Joshua Bloch, que fue Java Chief Architect en Google creo que responde a tu pregunta con mucha precisión, pues al ser entrevistado por Seibel dice “Even for problems that aren't inherently mathematical, the kind of thinking that you learn in math is essential to programming”. Estoy completamente de acuerdo. La matemática es una forma de pensar. La matemática no consiste en calcular. En cualquier libro de estudio del grado de matemáticas verás que el libro consiste en hacer demostraciones de teoremas y se hace mediante abstracciones y razonamiento lógico. Esto no tiene nada que ver con saber hacer una raíz cuadrada o hacer una integral definida. Los matemáticos usan muchas más letras que números y algunos de ellos se equivocan bastante al calcular. Es que calcular no es su negocio. Su negocio es demostrar. Las habilidades de un matemático para hacer razonamientos demostrativos con conceptos abstractos son muy útiles a la hora de programar, y por eso, depende de la matemática que te hayan enseñado puede que no te sirva para casi nada o te sirva de mucho. Si te han enseñado a demostrar teoremas y usar razonamiento abstracto entonces te están enseñando también “mathematical thinking” y eso es muy útil. Pero si te han enseñado las matemáticas que se enseñan en un grado de ingeniería (calcular) pues igual casi nada te es útil para programar.

Matizo lo anterior. Es cierto que hay cierto tipo de técnicas matemáticas que serán necesarias en función del tipo de programación que hagas. Por ejemplo, para los desarrollos actuales sobre IA con redes neuronales conviene dominar el álgebra lineal y los procesos de optimización (búsqueda de máximos y mínimos en una función). Para la IA que propone Judea Pearl conviene conocer la matemática discreta pues él plantea la búsqueda de la causalidad mediante redes bayesianas. De hecho incluso para hacer filosofía conviene, según Eric Steinhart, saber cierta matemática, como indica en su libro The math you need to do philosophy: conjuntos, máquinas de estados, probabilidad, conceptos sobre infinito y sobre complejidad.

Lo de pensar matemáticamente al programar no es ninguna tontería. Hay aproximaciones a la programación en el más puro estilo matemático-lógico. ¿Para qué? Pues para poder asegurar que un programa no contiene errores igual que te puedes asegurar usando las demostraciones matemáticas o lógicas que tus ideas se derivan lógicamente de los teoremas de partida.

Esta es la idea de la programación de Margaret Hamilton.

Ella fue la programadora del software con el que aterrizamos en la luna en 1969 y también la inventora del término “ingeniería del software”. Poca broma con Margaret Hamilton. Aún vive. ¿Por qué Margaret Hamilon quiere pensar la programación de forma lógica para evitar errores? Hay que recordar las dos alarmas que saltaron en el ordenador de a bordo del módulo de aterrizaje justo antes de posarse en la luna. Ella había programado ese ordenador para que el programa pudiera tomar decisiones en caso de conflictos. Y gracias a su forma de programar se salvó la misión porque el software resolvió por sí mismo dos errores. Esto la llevó a la investigación de formas lógicas de programacion, lo que ella llama USL (Universal Systems Language). Si le preguntas a Margaret Hamilton si es necesario saber matemáticas para programar seguramente te dirá que es necesario saber lógica matemática. Y para las cosas que ella hace tiene razón.

Y a los que usan el paradigma de programación funcional les viene bien conocer las abstracciones matemáticas porque los lenguajes funcionales las usan mucho.

Alguien puede pensar “pero yo no uso nada de eso, ni lenguajes funcionales, ni programo

compiladores, ni hago algoritmos de inteligencia artificial ni programo cosas gráficas". Pues entonces igual puedes pasar sin saber nada de matemática. Pero que sepas que lo que haces no es toda la programación aunque todas las personas que conozcas hagan lo mismo. Si no fuera porque hay gente que se plantea la programación matemáticamente muchos de los algoritmos o lenguajes que usamos no existirían. Conviene ser consciente de que el mundo es mucho más grande de lo que nosotros conocemos y la informática no es una sola cosa, como la filosofía tampoco es una sola cosa. También hay "informáticas". Hay mucha ciencia en la informática y también hay muchas técnicas. Si te ha tocado estar del lado de las técnicas y aprendiendo patrones de programación te puedes ganar la vida pues estupendo. Pero hay gente que trabaja en el lado científico de la informática y a esa gente las matemáticas les vienen muy bien. Cuando vas a la Universidad nadie sabe de qué lado vas a caer (puramente técnico o científico) y supongo que es por eso que te enseñan matemáticas. Para que puedas caer en el lado científico de la informática. Quiero pensar que es por eso.

Entonces, respondiendo a tu pregunta: ¿hace falta saber matemáticas para programar? contestaría que depende. Estorbarte no te va a estorbar. Y la forma de pensar matemática te puede hacer mucho más fácil la vida al diseñar sistemas de información.

Añado un dato como curiosidad.

También el "mathematical thinking" les ha sido útil a algunos filósofos a la hora de plantear nuevos "frameworks". Por ejemplo, el conocido filósofo Spinoza tiene un libro titulado "Ética demostrada según el orden geométrico" en el que elabora una argumentación en el más puro estilo matemático, con definiciones, axiomas, postulados y proposiciones… ¡para hablar sobre ética! Te explota la cabeza al leer ese libro. Es de una originalidad impresionante. Es un ejemplo de cómo la forma de razonar matemática es útil fuera de la matemática, de hecho en un terreno muy alejado de lo que suele estar matematizado: la ética.

La siguiente pregunta es de un lector que conoces, José Carlos García, y, aprovechando su pregunta, se me ha ocurrido otra similar.

La pregunta de José Carlos: ¿Cómo propondrías la enseñanza de la filosofía en los institutos?

Y mi pregunta añadida: ¿Cómo propondrías la enseñanza de la programación en los institutos?

¿Cómo propondrías la enseñanza de la filosofía en los institutos?

Me haces una pregunta sobre un asunto del que no tengo ninguna experiencia: la enseñanza de la filosofía a adolescentes. Toda mi experiencia docente ha sido con adultos así que no puedo aportar nada que proceda de mi experiencia.

Sobre los programas oficiales que se usan como guion para las clases de filosofía sí te diré que el programa para el primer curso de bachillerato no está mal, aborda la filosofía por temas y hay temas bastante actuales, lo cual permite conectar la filosofía con el presente, que, como sabes, es la

orientación que prefiero. Creo que al temario de filosofía de primero de bachillerato se le puede sacar mucha punta llevando a la clase lo que sucede en el presente y haciendo que los alumnos puedan opinar y debatir. Me consta que esto se hace así por muchos profesores. Un gran aplauso a ellos.

En cambio el programa para segundo de bachillerato, la historia de la filosofía, me parece bastante rancio la verdad. Pongo un ejemplo. Al explicar la filosofía de la Edad Media se centra en Tomás de Aquino y el agustinismo y, por tanto, el alumno se pierde todos los debates filosóficos medievales que hicieron posible el nacimiento de la revolución científica matematizada en el siglo XVII. Tal como se cuenta la historia del pensamiento parece que en la Edad Media estaba todo el mundo hablando de religión y de repente surgen Galileo y Newton contando algo diferente como caídos del cielo y empieza la razón científica. Es tan ridículo que hace hasta gracia. Propondría denominar a esa asignatura “Historia del pensamiento” y cruzar el desarrollo de la historia con las novedades del pensamiento de cada época. Cuando se aísla un discurso filosófico de su contexto histórico se vuelve ininteligible.

Hay profesores de filosofía que están haciendo cosas muy interesantes, como Eduardo Infante con sus #FiloRetos en Twitter y su libro “Filosofía en la calle” que me parece una forma muy interesante de conectar la filosofía con la realidad cotidiana. Sé que hay muchos profesores que tratan de innovar en la forma de enseñar la filosofía y tienen mucho mérito por ello.

¿Cómo propondrías la enseñanza de la programación en los institutos?

Sobre esta pregunta tengo menos idea aún. No estoy informado del plan de estudios sobre programación o informática en los institutos. No creo que pueda hacer ninguna propuesta a profesionales que tienen mucha más experiencia que yo.

Sin embargo sobre esto tengo una mínima experiencia que puedo compartir por si se puede aprender algo de ella. Me refiero a mi estancia de un mes en Nepal en 2008, donde estuve dando clase de informática a chavales de Grade-6 (alrededor de 11-12 años) y de Grade-9 (alrededor de 14-15 años). Me compré los libros de texto de esos niveles y vi que los libros tenían cosas interesantes y aprovechables aunque en Grade-9, donde enseñaban programación, el lenguaje que les enseñaban era QBasic. Aquello no pintaba bien.

El director de la escuela me dijo que él consideraba muy importante que los alumnos aprendieran programación de ordenadores porque la programación enseña a tener disciplina mental y eso es aplicable a cualquier cosa que hagan aunque no sea propiamente informática. Y tenía razón. Por eso me dio libertad para que planteara las clases como yo quisiera pero tomando en consideración que lo que aprendieran les pudiera servir aunque no se dedicaran a la informática. Me pareció un planteamiento muy inteligente y así lo hice.

Plantee las primeras clases como un ejercicio de análisis de una actividad cotidiana, descomponerla en partes y escribir eso en un inglés lo más formalizado posible. Lo hice así porque así aprendí yo mismo a programar: con papel y lápiz. Recuerdo que usamos el proceso de lavarse la cabeza con champú y tuvimos que analizar en qué consistía el proceso y nos “inventamos” un lenguaje formal para escribirlo. Hubo debates sobre si ir a comprar el champú debía de estar incluido en el proyecto o si sacar el champú del armario del baño sería un programa aparte. Vimos también que abrir un tapón de rosca (el de la botella de champú) podría ser un proceso reutilizable en otros programas y lo separamos. Hicimos un proceso principal que llamaba a todos esos subprocesos: comprar champú si

no tenemos, abrir el armario para coger el champú, etc. Aprendimos también a hacer procesos iterativos, bucles: poner champú, restregar, aclarar. Y así, con un ejemplo cotidiano, fueron captaron las ideas de cómo pensar analíticamente. Fue muy divertido plantearlo así.

Como en 2007 acababa de lanzarse un producto del MIT (Massachusetts Institute of Technology) para enseñar a niños a programar, un programa llamado Scratch, que ahora es bastante conocido, decidí usarlo. Cuando empezamos a usar Scratch para aplicar lo que habíamos aprendido de forma teórica en las “clases de champú” hicieron verdaderas maravillas con él en las pocas semanas que estuvimos usándolo. Incluso varios videojuegos básicos. Fue un mes muy intenso y conseguí crear varias vocaciones informáticas. Varios de los alumnos que tuve han estudiado informática y ya están trabajando profesionalmente como desarrolladores. Sigo manteniendo contacto con ellos.

No sé si todo eso que viví puede servir para contestar a tu pregunta o es tan solo una experiencia de interés local, pero me parece que hubo muchos elementos que siguen de actualidad: temarios atrasados, directores innovadores que piensan a lo grande, profesores innovadores y chavales que responden cuando el profesor se emociona de lo que explica. Creo que la emoción es un vial por el que se pueden transmitir los conocimientos muy eficazmente y se pueden crear vocaciones.

¿Será algo de esto aplicable a la programación en los Institutos ahora? Pues no lo sé. El sistema es muy diferente, la actitud de los alumnos hacia el profesorado es muy diferente (allí era casi adoración la que los chavales sentían por los profesores, no solo por mí) y puede que la libertad que me dieron no fuera habitual, no lo sé. Pero lo cuento por si esto sirve para hacer una reflexión sobre lo que se está haciendo ahora.

Uno de los usos de la IA es su utilización en los robot, que ahora mismo son poco mas que una “herramienta inteligente” para realizar determinadas tareas. Pero en diversos libros, series o películas se trata a los robot como una copia mejorada de un ser humano. De hecho tenemos las categoría de “robot humanoide” que pretende simular el comportamiento y los movimientos de humano. Se podría considerar que los movimientos de un ser humano son un proceso de ingeniería pero el comportamiento es mas programación de un IA. Ese comportamiento implica muchas cosas como la creatividad, imaginación o sentimientos. En la actualidad hay proyectos que ya intentan desarrollar esos comportamientos, tales como aplicaciones que componen música.

¿Se puede programar la creatividad, imaginación o los sentimientos en un IA?

En el episodio 11 de nuestro podcast titulado ¿Qué es la creatividad? citamos las filosofías info-computacionales, el trabajo de Jurgen Schmidhuber y un trabajo en la carrera de filosofía en el hablé sobre su investigación, que tiene que ver con la intersección entre la creatividad y la computación. Creo que esta cita de ese trabajo que usé en el podcast contesta tu pregunta:

”Jürgen Schmidhuber en su trabajo Simple Algorithmic Principles of Discovery, Subjective Beauty, Selective Attention, Curiosity & Creativity postula un funcionamiento de la inteligencia según la cual el agente captura y almacena las observaciones sensibles que llegan a sus sentidos y, en un momento determinado, dadas las capacidades computacionales del agente, una parte de esos datos sensoriales almacenados en crudo es reconocida como comprimible según un programa interno que constituiría lo que llamamos descripción, explicación o modelo del mundo. Subjetivamente dicho agente considera que tales datos cuya regularidad permite su compresión son más bellos que los demás datos no comprimibles. El hecho de que el agente sea más competente en aplicar compresión a un conjunto de datos procedentes de las observaciones sensoriales da comienzo a un círculo virtuoso al que Schmidhuber denomina interestingness, haciendo así del interés y la curiosidad la primera derivada de la belleza subjetiva, pues es el indicador del cambio en la compresión que se ha producido.“

Según este enfoque filosófico la creatividad, la comprensión intelectual y la apreciación artística serían modos diferentes de la misma capacidad: la capacidad de reducir la complejidad de los datos en crudo que nos llegan del mundo. Es decir, que siguiendo por esta línea quizás podríamos averiguar si esas capacidades tan elevadas son exclusivamente humanas o son algo que surge en la naturaleza si se dan ciertas circunstancias.

Por otra parte hay que tener cuidado contra lo que se llama el prejuicio substratista, es decir, la idea de que para que algo no humano tenga estatus moral tiene que ser lo más humano posible. Este tipo de prejuicio de “las máquinas no son inteligentes/creativas/sensibles porque no son como nosotros” nos encierra dentro del ámbito humano y nos hace poseedores exclusivos de “lo elevado” porque la definición de “lo elevado” necesita estar dentro de un ser que sea humano. Un argumento circular. Sustituye “lo elevado” por creatividad, emociones, capacidad moral y ya tienes un conjunto de problemas filosóficos que estudiar. El filósofo Zachary Biondi es una referencia en esto y hablaremos de él en algún episodio próximo del podcast que aún está en preparación. El tema que trataremos será el del estatus moral de las máquinas.

¿Crees que es necesario que una IA tenga esos comportamiento humanos?

Aquí ya hablas de “comportamientos humanos”. Esto es otro tema. Esto es parecer humano. Yo creo que está claro que si un robot despliega comportamiento humano hará que tengamos de manera natural una empatía hacia él. Nos pasa igual con los animales. En el comportamiento de los perros vemos cosas como la fidelidad y la amistad y por eso decimos que el perro es nuestro mejor amigo y, sin embargo, no vemos eso en otros animales y, por tanto, no desplegamos tanta empatía hacia ellos.

En su libro Machines Who Think (libro recomendadísimo) la historiadora de la Inteligencia Artificial Pamela McCorduck habla de lo que ella llama un “geriatric robot”, una máquina que podría hacerse cargo de la atención de personas mayores, especialmente de tareas que requieren fuerza física como incorporar a enfermos o tareas semejantes. Lo que ella propuso en ese libro es un experimento mental, pero si tal robot existiera algún día (y en Japón van en esa línea) evidentemente sería mejor que pudiera tener un trato con nosotros que reconozcamos como amable y educado. Recuerda los robots de la película Interstellar, Tars creo que se llamaba uno de ellos, que tenían un sentido del humor regulable a voluntad para poder interactuar con los seres humanos de una forma más natural para nosotros.

Sí, definitivamente el interfaz de los robots o de cualquier máquina debe de estar adaptado a

nosotros, debería de tener “comportamiento humano”, para que podamos interactuar con naturalidad con ellos. Y eso es independiente de que sean o no sean humanos. Una cosa es comportarse como humanos y otra es ser humanos, al test de Turing me remito. Y, a veces, con un comportamiento humano podemos tener suficiente.

Esta es la pregunta menos tradicional que te voy a hacer que realmente no se como definirla. Simplemente te propongo una situación donde debes tomar una decisión y la pregunta es sobre esa decisión. A continuación la situación.

Eres el encargado de administrar y almacenar todo el conocimiento de la humanidad en un recinto que tienes organizado en diferentes materias: música, matemáticas, física, filosofía, arte, historia, leyes, medicina…etc. Todo el conocimiento recopilado por el ser humano a lo largo de toda su historia.

Ese recinto sufre un gran desastre que provocará la pérdida de todo el conocimiento humano aunque todavía tienes tiempo a salvar UNA SOLA materia que te debería de permitir, en un futuro, reconstruir el conocimiento humano perdido de la forma mas rápida posible.

¿Qué materia salvarías?

¿Qué materia salvarías?

Si la filosofía, según su etimología, es el amor por el conocimiento y de la filosofía han surgido las ciencias especializadas entonces salvando la filosofía salvas el amor por saber y una forma de conocimiento que está conectada con todo. Salvando la filosofía salvas la fábrica de nuestras herramientas de conocimiento. Voy a argumentarlo.

La filosofía no es una profesión, es una forma de acercarse al conocimiento, es un “mindset”, un “framework”, esquemas mentales, métodos, es un arsenal de metodologías entre las que se pueden incluir la especulación y la lógica. Yo lo veo gráficamente así: si pintas unos ejes cartesianos y pones en el eje de abscisas (eje x) todas las ciencias y conocimientos especializados desarrollados por los seres humanos puedes encontrar un número en el eje de ordenadas (eje y) que corresponda al número de escritos filosóficos sobre ese tema. Creo que no quedaría excluido ningún tema.

La filosofía nace en Jonia y Grecia como una tercera vía para el conocimiento entre dos formas establecidas de conocimiento a las que desafía: la mera opinión cambiante de los hombres y el decreto autoritario de los dioses. Y elabora una forma nueva de acercarse al conocimiento, un método racional que incluye la lógica, la matemática y la especulación. Si lees la Metafísica y la Lógica de Aristóteles te das cuenta de que allí ya está escrita nuestra metodología para conocer el mundo.

Pero la filosofía no está conectada solo con las ciencias. También lo está con las artes. La estética es una rama enorme de la filosofía y apasionante. Es una reflexión sobre nuestra capacidad de conectar

con la realidad y también sobre nuestra capacidad de aprecio artístico. Hay toneladas de escritos filosóficos sobre ello. Quiero recordar de nuevo que en nuestro podcast hemos hablado de la relación entre la filosofía y la danza. Y quiero recordar que la música es una disciplina que se estudia matemáticamente desde Pitágoras y Platón heredó todo eso. Es precisamente la estética la que nos permite introducir la computación en los terrenos filosóficos mediante la infocomputación. La importancia de la literatura quedaría salvada con filósofos como Richard Rorty, que ha hecho del valor de la literatura su propuesta filosófica. Pero hay mucha filosofía de cosas inimaginables, del deporte, del universo, de la física, del cuerpo, de los videojuegos, de la tecnología, de la información. Salvando la filosofía le volverían a nacer "hijos" que se independizarían y volverían así a surgir las ciencias especializadas. Salvando a la madre vuelves a tener descendencia.

La matemática sería una gran candidata a ser esa materia que podríamos salvar para reconstruir el conocimiento ya que es una disciplina también básica y que hemos sido capaces de conectar con todos los saberes. Pero esa concepción de la matemática como arma de conocimiento se propuso por primera vez en Grecia y fueron los filósofos los que lo hicieron. Como comenta el profesor de matemáticas Keith Devlin en su "Introduction to Mathematical Thinking" fueron los griegos los que llevaron más allá la matemática de los Babilonios y Egipcios. En Grecia la matemática dejó de ser una colección de técnicas para medir, contar y contabilizar y pasó a ser un área de conocimiento. Fue el filósofo Tales de Mileto quien por primera vez introdujo la idea de que las afirmaciones matemáticas podían ser probadas mediante argumentos formales. Boom. Gracias a esto que hizo Tales la matemática deja de ser una herramienta que solo sirve para contar y como consecuencia aparece el concepto de teorema. Y entonces llega Euclides y escribe sus "Elementos" y el resto es historia. Todo lo que toca la filosofía queda convertido en un arma de conocimiento.

Comentas en tu pregunta diversas especialidades científicas muy importantes, pero esta forma de ver el conocimiento, compartimentado, como división por especialidades, es muy reciente. Es consecuencia del positivismo lógico de Augusto Comte y su pirámide del saber. Antes de eso los filósofos (y sigue ocurriendo) estaban metidos en todos los charcos, sean estos científicos, éticos o estéticos. No hay área de conocimiento ajena a la filosofía. Unos ejemplos conocidos. En la entrada de la Academia de Platón había una inscripción que decía "No entre aquí nadie que no sepa geometría". Leibniz escribió una teoría de la complejidad, desarrolló el cálculo infinitesimal y habló sobre el problema de la maldad en el mundo. Descartes inventa la geometría analítica y da comienzo a la filosofía moderna sin sentir que estaba haciendo algo diferente. No hay nada desconectado de la filosofía y todo lo que ella toca lo convierte en un arma de investigación. Por eso habría que salvarla.

La última pregunta de esta serie no es mía, es de un entrevistado al que pedí que planteara una pregunta a un desarrollador, aunque la pregunte a un entrevistado anterior, pero no respondió. Sé que tu perfil no es desarrollador, pero la temática de la pregunta creo que encaja contigo y trata temas que ya has respondido en esta entrevista.

¿Cómo ves que tu carrera esta cambiando y cambiará con el dominio cada vez mas sólido de la inteligencia artificial?

Pues en mi trabajo, que consiste en sysadmin de sistemas, redes y servicios, creo que aún estamos

es la puerta de entrada de la inteligencia artificial, es decir, en la automatización. Hemos conseguido convertir en virtuales tanto los servidores como las redes y ahora podemos desgajar fácilmente aplicaciones de un servidor, virtualizarlas con todas sus dependencias usando las técnicas de contenedores y pasarlas a otro servidor más conveniente. Con los orquestadores inteligentes que ya tenemos, que nos ayudan a usar esa automatización, podemos hacer hoy mismo grandes cosas para que los propios sistemas tomen decisiones que garanticen la continuidad de los servicios. Pero eso no es inteligencia artificial. Eso son buenos algoritmos.

Dotar de más “inteligencia” a ese orquestador, a la capa de control, y que sea capaz de adelantarse a los problemas sería muy deseable pero con los modelos de inteligencia artificial actuales basados en Big Data no veo que podamos llegar muy lejos en esta dirección aunque claramente hay casos en los que se podría usar y mucho.

Otro gallo nos cantaría si la inteligencia artificial aplicable a los sistemas informáticos estuviera basada en contrafácticos en vez de en Big Data. En ese caso el sistema “comprendería” mis necesidades y sabría qué hacer sin necesidad de correlar mis necesidades con un Big Data de otras instalaciones informáticas que vaya usted a saber si son representativas de lo que a mí me sucede o si la forma de resolver estos problemas, según se deduce de esos datos, es aplicable a mi instalación.

En mis side projects de programación sí que he tenido oportunidad de usar IA para reconocimiento de lenguaje natural escrito en el caso del chatbot para el ensayo clínico del que ya hablé anteriormente.

En mi carrera filosófica sí que ha afectado el asunto de la inteligencia artificial, de hecho incluí el tema en mi Trabajo de Fin de Grado, como ya comenté. Y en el podcast “La filosofía no sirve para nada” seguimos de cerca los debates filosóficos sobre el asunto. Como dijo el filósofo Max Horkheimer cuando le preguntaron para qué sirve la filosofía, “la filosofía se preocupa de que no nos timen”. No solo que no nos timen los demás, también se trata de que no nos timemos a nosotros mismos. Por eso cuando algo se me presenta como “Inteligencia Artificial” procuro intentar que no me timen, procuro entender su funcionamiento para ver si está justificado ese nombre y procuro conocer alternativas a lo que se me está vendiendo cómo única solución. Me remito a Gary Marcus, del que ya hablé aquí, y su idea de mezclar el enfoque simbólico de la inteligencia artificial con el enfoque basado en redes neuronales.

Este es un tema que está en pleno crecimiento y todo parece indicar que nos afectará una manera o de otra. Habrá que estar preparado, es decir, habrá que estar dispuestos a cambiar pero no dispuestos a cualquier cambio.

¿Qué te hubiera gustado que te preguntase? Evidentemente debes responder a tu propia pregunta.

Pues una pregunta que yo me he hecho en varias ocasiones: si volviera a nacer ¿volvería a ser informático y filósofo?.

Mi respuesta actual a esa pregunta es que si pudiera recordar mi vida anterior entonces no repetiría, haría cosas distintas y creo que sería matemático y músico.

Pero si comenzara desde cero y no recordara nada de una vida anterior entonces creo que volvería a repetir. La programación, la comunidad y los valores del mundo hacker y todos los mundos a los que he tenido acceso a través de la filosofía me volverían a seducir y volvería a caer en sus redes. Con toda seguridad. No se me ocurren dos mejores temas a los que dedicar una vida.

Pero como no creo que haya otras vidas después de esta pues procuro hacer hueco ahora a esos otros temas que me apasionan. Leo mucha historia de la matemática y estudio la que puedo. Y estudio teoría musical aplicada al piano. Aunque en estos dos temas soy un modesto aficionadillo me causa mucho placer lo poco que sé.

Por último, puedes indicar tus métodos de contacto y si tienes algún proyecto, web, podcast o evento que quieras promocionar tienes este espacio disponible.

Gracias a ti por tu invitación, me ha parecido una experiencia muy estimulante.

Me pueden contactar por Twitter a mi cuenta personal https://twitter.com/joakinen o a la cuenta del podcast en Twitter, https://twitter.com/FilosofiaNada.

Si les apetece se pueden pasar por la web filosofias.es donde cada episodio del podcast tiene su correspondiente cuaderno de notas. Estos son los episodios grabados hasta la fecha: https://filosofias.es/wiki/doku.php/podcast/episodios

También tenemos un grupo en Telegram con gente a la que le gusta la filosofía y donde hablamos de todo tipo de temas procurando sacarle el lado filosófico: https://t.me/opinafilosofianada

Propongo para una futura entrevista a Juan Antonio Torrero, en Twitter https://twitter.com/jatorrero

Sobre qué pregunta te propongo para un sysadmin: ¿qué opinas sobre la relación actual de Windows con Linux y la decisión de integrar el kernel de Linux dentro de Windows como un subsistema más, el WSL2?

Un placer. Hasta siempre.