La macchina che si programma da sola: perché il futuro sarà alla portata di tutti

di Francesco Maria De Collibus

20-05-2025

La società e l'economia contemporanee sono dominate dal software. Sette delle dieci persone[1] più ricche del mondo hanno o hanno avuto a che fare con il software. La ricchezza non farà forse la felicità, ma resta uno dei più concreti indicatori di qualcosa altrimenti difficile da misurare come il successo. Fortune senza precedenti nella storia umana sono state realizzate attraverso il software, parliamo di aziende come Google, Microsoft, Oracle e Facebook. Persino Elon Musk, la cui immensa fortuna adesso pare legata a prodotti più tangibili come i razzi e le automobili, è partito da una azienda di software, Paypal. Già nel 2011 Marc Andreessen[2] ci avvertì che il software si era «mangiato il mondo» e negli ultimi anni la sua insaziabile fame non sembra certo essersi placata. Nonostante la sua importanza e pervasività, il software resta per molti qualcosa di sfuggente e indecifrabile. L'aggettivo soft soffice, presente nella parola composta è ingannevole: il software più che soffice è forse inafferrabile. Probabilmente una delle ragioni principali per cui non capiamo il mondo di oggi e a volte questo ci appare assurdo nelle sue dinamiche è che non capiamo il software, sappiamo che esiste e che è importante, ma non ne abbiamo un modello mentale efficace. Non comprendiamo la sua fragilità e la sua efficienza, non capiamo come faccia a smuovere con efficienza le montagne, per poi inciampare con goffaggine sul primo sassolino, con effetti catastrofici. Il software muove silenziosamente ed efficientemente il mondo, però poi un bel mattino cambia una riga della configurazione di un provider sconosciuto ai più e il tuo aereo non parte più per giorni, come è successo la scorsa estate con il caso Crowdstrike[3]. Non c'è settore che non possa essere colpito da problemi con il software: treni fermi perché la centralina che li comandava è stata staccata dalla corrente[4], ospedali che non possono più accedere alle cartelle dei pazienti a causa di un attacco ransomware da parte di cybercriminali[5], l'infrastruttura finanziaria del Paese che di colpo non accetta pagamenti digitali[6] e così via. A volte il software non solo muove valuta, il software stesso si fa valuta, il software diventa il denaro stesso come avviene nelle criptovalute, in una sorta di trasmutazione alchemica verso l'oro che dice molto della nostra società contemporanea[7]. Non avendo una componente materiale da produrre e distribuire, il software "scala" in maniera molto efficace, senza essere legato a componenti materiali quali produzione e distribuzione, e i mercati finanziari apprezzano questa scalabilità rapidissima e la premiano generosamente in borsa. Persino l'articolo che state leggendo in questo momento è stato scritto tramite un software di videoscrittura, inviato via mail tramite un web browser, e reso disponibile al pubblico tramite una applicazione web, oppure impaginato e stampato su carta sempre tramite altri programmi. Ogni singola attività del mondo di oggi richiede software, molto software. Ormai ci siamo assuefatti a tutto questo, ma se vi fermate a pensarci è qualcosa di straordinario e sbalorditivo.

Come ricorda Cosimo Accoto[8] ormai qualsiasi organizzazione nel mondo produce soprattutto software. Nelle sedi centrali di banche, assicurazioni, industrie alimentari, imprese farmaceutiche, i bancari, gli attuari assicurativi, gli esperti di prodotto o i ricercatori farmaceutici si trovano spesso in minoranza, sovrastati numericamente da reparti IT (Information Technology) in crescita e sempre con posizioni aperte: l'innovazione ormai passa in buona parte nelle loro mani, portando nuovi modi di lavorare e di processare le informazioni. Quasi tutte le grandi organizzazioni aziendali e governative ragionano con i tempi del software, con rilasci, cicli e aggiornamenti. Questo sia nei

propri strumenti di operatività interna sia nei prodotti e servizi che rilasciano all'esterno: la sostanza non cambia.
Si potrebbe pensare che questi ben assortiti reparti informatici abbiano perfettamente sotto controllo il software. Sono informatici, scienziati no? In realtà fino a un certo punto. I reparti IT di tutto il globo cercano in qualche modo di tenere a bada un mondo molto complesso, con cambiamenti tecnologici tanto frequenti quanto radicali. Il modo in cui questi volenterosi impiegati tengono a bada la esponenziale complessità della propria disciplina è tanto variopinto quanto ancora sorprendentemente artigianale. Ci sono numerose tendenze del momento, dove fantasiose metodologie dal successo empirico vengono introdotte con un ritmo più o meno costante. Da metodologie "storiche" come il Waterfall, la cascata, proposta sin dagli anni Settanta, si è passato al lavoro di luminari dell'ingegneria del software come il Rational Unified Process[9], fino a inventive metodologie "agile" come lo Scrum, il Kanban, con strumenti come Planning Poker, una sorta di gioco di società dove i membri del team di sviluppo "scommettono" su quanto sarà difficile implementare una determinata feature nel prodotto e fanno così previsioni temporali sul prossimo futuro. Come tutte queste metodologie, fondamentalmente funzionano finché funzionano e finché non si trova qualcosa di meglio.
La disciplina dell'ingegneria del software si è sempre trovata un po' a disagio nel confronto con le altre discipline ingegneristiche tradizionali. Se paragonassimo i risultati raggiunti dal software con quelli, per esempio, dell'ingegneria meccanica o civile, avremmo automobili che vanno a 40.000 km/h ma che improvvisamente esplodono durante un parcheggio, oppure ponti costruiti in due giorni che reggono il carico di decine di camion e treni, ma crollano di colpo quando una bicicletta verde prova ad attraversarli, o un ponte che va smontato e ricostruito completamente ogni cinque anni. I programmatori non hanno ancora del tutto deciso se il proprio mestiere sia effettivamente una scienza o piuttosto un'arte: per questo molti preferiscono attribuirsi nuovi e accattivanti titoli professionali come "Software Craftsman". Forse è sia scienza che arte, di sicuro resta imperscrutabile, tanto è vero che uno dei libri più interessanti sulla pianificazione del software si intitola Software Estimation. Demystifying the Black Art [10]. Quanto ci vorrà a sviluppare qualcosa? In realtà lo sa solo il buon Dio, ma siccome siamo in un'epoca secolarizzata, non vogliamo ammetterlo. In ogni caso le aziende moderne vivono di pianificazione e consuntivi, quindi qualcosa bisognerà pur dire. E se anche con un misto di fortuna e istinto riuscissimo a consegnare nei tempi che avevamo previsto? Il software va comunque collaudato ("testato"), aggiornato, spiegato agli utenti finali e dismesso alla fine del suo ciclo di vita. Una regola empirica stima che il 60% dei costi legati a un progetto software non sia dovuto al suo sviluppo, ma al ciclo di vita della sua manutenzione. Il software per queste ragioni è una risorsa scarsa, non si può scrivere software per tutto, sarebbe troppo costoso farlo: il software viene scritto quando ha senso automatizzare o gestire una situazione in modo automatico.
Ma se anche troviamo le ingenti risorse e le carenti figure professionali per realizzare la nostra benemerita applicazione? Una delle più grandi cause di frustrazione per un utente di un sistema informatico è che il computer notoriamente "fa quello che gli dici, non quello che vuoi". Nelle interfacce utente tradizionali questa discrepanza tra intenzioni e risultati ottenuti può essere notevole, causando un impegnativo lavoro di ricerca da parte di una serie di figure professionali nuove come gli esperti di User Experience (UX), i quali si prodigano per evitare le sfuriate e i travasi di bile dei malcapitati utenti. Il computer non può, o meglio non poteva, semplicemente ascoltarti, il comando deve essere impartito in un modo che la macchina riesca a capire. Questo vuol dire cliccare sull'icona giusta, inserire l'input corretto nel campo richiesto. Una password non è quasi giusta, o lo è interamente o è sbagliata. All'input corretto deve seguire l'output corretto, ogni deviazione va considerata come un errore del programma, o un bug da correggere.
Tutte queste assunzioni sono mutate da quando sono disponibili i Large Language Models. Per anni

discipline come la linguistica computazionale o il NLP (Natural Language Processing) si erano impegnate per rendere accessibili ai calcolatori linguaggi naturali con i quali noi esseri umani ci esprimiamo con tanta scioltezza. Le macchine gestiscono molto bene i linguaggi formali e ben specificati (ad esempio un linguaggio di programmazione), ma hanno (o meglio, avevano) problemi a comprendere frasi irregolari e ricche di sfumature come quelle con cui abbiamo il vizio di esprimerci noi umani. L'assegnazione di un valore numerico a una variabile è semplice. Le ambiguità di un periodo ipotetico, con le sue sfumature lessicali lo sono molto meno: la macchina suda per processare il nostro linguaggio. Questa è una metafora, una cosa che una macchina con un insieme fisso di regole avrebbe difficoltà a capire. Ma se chiedo lumi a riguardo a ChatGPT (modello o1): «Se affermi che una macchina "suda" per elaborare il linguaggio naturale umano, stai utilizzando una metafora per indicare la difficoltà, lo sforzo o la complessità del processo. Una macchina, in realtà, non può sudare come un essere umano, ma puoi immaginare il "sudore" come il dispendio di risorse computazionali, l'aumento del carico di lavoro o la necessità di algoritmi avanzati e di una considerevole potenza di calcolo. In altre parole, dire che una macchina "suda" significa che il compito di comprendere, analizzare e generare il linguaggio umano è impegnativo e richiede molta "fatica" dal punto di vista del calcolo».

Non si sa con precisione perché gli approcci GPT (Generative Pretrained Transformer) si siano rivelati così efficaci: eppure superata la soglia di un certo numero di parametri durante l'addestramento, sono apparse come proprietà emergenti[11] non solo la comprensione del linguaggio, ma anche molte altre abilità specifiche del tutto inattese, dal riassumere documenti al fare battute. I vari modelli GPT fino al 3 si erano rivelati promettenti per gli addetti ai lavori, ma non così sorprendenti per gli utenti finali. I computer in grado di parlare come noi erano stati oggetto di fantascienza: tra i moltissimi esempi, vengono in mente HAL-9000 di 2001: Odissea nello spazio, o Teletraan-1, il computer degli Autorobot nel cartone animato dei Transformers. Immaginavamo che un giorno i nostri figli e nipoti avrebbero potuto parlare con le macchine come noi parliamo tra di noi, non certo di poterlo fare noi stessi dopo appena qualche anno. Anche se approcci di questo tipo di Voice Recognition, come Apple Siri o Amazon Alexa, erano già stati commercializzati da tempo, questi presentavano comunque limiti ben evidenti e si limitavano per lo più alla comprensione di comandi piuttosto basilari. L'idea di poter semplicemente spiegare a una macchina quello che vogliamo senza limiti di significato ottenendo risposte pertinenti era consegnata appunto alla fantascienza. Adesso invece la macchina ci comprende molto bene, anche con le nostre sfumature di significato.

Le macchine sembrano aver improvvisamente ricevuto il dono del linguaggio, e adesso con Riccardo Manzotti[12] e David J. Chalmers[13] ci chiediamo se possano ricevere anche altri doni, come per esempio la coscienza. Eppure, le macchine (o almeno gli elaboratori elettronici) hanno sempre posseduto il linguaggio, solo che il loro era un linguaggio formale, ben specificato. Adesso parlano la nostra lingua e sembrano capire senza problemi un testo anche con errori di ortografia, errori che una volta avrebbero portato alla più totale e sorda incomprensione. Adesso invece tutta una serie di nuove possibilità, ruoli e funzioni è stata resa accessibile alle macchine: per essere indirizzati al servizio giusto di un centralino telefonico prima dovevamo schiacciare l'apposito tasto, altrimenti il software del call center non ci poteva comprendere, adesso possiamo semplicemente spiegare le nostre necessità come le spiegheremmo a un impiegato umano. C'erano state, anche prima degli LLM, avvisaglie in questo senso, da Microsoft Bot Framework a Google Dialogflow, ma nulla di paragonabile a quello che abbiamo oggi.

Questa capacità - sempre più raffinata con il succedersi di modelli sempre più potenti - è talmente sconcertante che ha oscurato una capacità altrettanto sorprendente che i Large Language Models possiedono: possono scrivere molto bene programmi. Riescono a capire quello che vogliamo dalle nostre espressioni in linguaggio naturale e tradurlo nel linguaggio formale di un software. Per ironia

della sorte, uno delle prime categorie professionali a cui l'Intelligenza Artificiale rischia di togliere il lavoro sono proprio quei programmatori che tanto hanno contribuito negli anni per eliminare il lavoro altrui. In realtà non sarà così, persone competenti in materia continueranno a servire[14], ma la verità è che anche modelli LLM molto compatti riescono a scrivere codice in modo efficace.
Non si parla solo di codice: gli LLM eccellono in molti compiti di traduzione da specifiche in linguaggio naturale a specifico linguaggio formale ben definito. Durante i primi giorni dei generatori di immagini tramite IA ci si doveva arrovellare per trovare il "prompt" giusto che avrebbe potuto generare l'immagine desiderata, problema che già con i generatori attuali appare superato. Fu una scoperta sapere che bastava spiegare il problema a ChatGPT per ottenere il prompt giusto. Scrivere software potrebbe smettere di diventare un'impresa titanica riservata agli specialisti, per diventare qualcosa alla portata di tutti. Il software è il sistema nervoso della nostra società, e permette a istituzioni, aziende e cittadini di funzionare e interagire nello spazio sociale. Non solo il software automatizza e velocizza i processi ma permette anche di creare ed estrarre dati in maniera efficiente. Il software scrive dati, i dati vengono letti e di qui può partire l'ottimizzazione dei processi stessi, come la gestione di un magazzino, la prenotazione degli appuntamenti e così via. Sarebbe possibile estrarre dati anche senza software (la statistica come disciplina precede di molto l'apparizione del computer), ma mai nella quantità e qualità che ci permette di ottenere l'informatica. Con i dati si dovrebbe essere in grado di prendere decisioni migliori: software e dati sono in un circolo virtuoso, gli uni abilitano gli altri, in settori sempre nuovi. I dati scritti dai programmi sono il carburante delle applicazioni più raffinate della Data Science o del Machine Learning. Il software potrebbe presto cessare di essere una risorsa scarsa, tutti potremmo essere in grado di scrivere rapidamente tutto il codice che ci serve, o di pensare a nuovi casi di uso, e innovare in prima persona portando realizzando idee nuove. Oggi milioni di persone nel mondo lavorano nel campo del software con una efficienza spesso difficile da misurare. A volte centinaia di sviluppatori producono blandi aggiustamenti a un'applicazione vecchia di anni, altre volte una manciata di sviluppatori motivati producono dal nulla qualcosa in grado di sconvolgere il mercato, come è successo agli albori con Whatsapp, Instagram, Bitcoin o Ethereum. È la "magia nera" di cui si parlava prima. Ma mentre prima scrivere software era una impresa estremamente difficile, adesso tramite strumenti come Github Copilot, Replit, Cursor.ai, Amazon CodeWhisperer, Google Gemini Code Assist e numerosi altri è diventato possibile sia agli specialisti scrivere più rapidamente codice che ai non specialisti scrivere programmi. L'approccio low-code, o addirittura no-code, scrittura di programmi senza alcun bisogno di programmare, ha permesso di ridurre di molto la quantità di codice che bisogna scrivere per ottenere una applicazione funzionante, ma - prima degli LLM - restavano casi di nicchia o molto limitati nei risultati raggiungibili. Adesso si può aprire un editor e semplicemente descrivere al Large Language Model quello che si vuole fino a ottenerlo. Ovviamente bisogna ancora avere competenze tecniche per valutare sia le domande da fare che la qualità delle risposte ottenute, e soprattutto per mettere insieme i componenti della soluzione. Il processo di scrivere codice rischia di diventare una pratica inaspettatamente "socratica", fondata sulle domande da fare e il percorso di una discussione. Si parte scrivendo qualche linea di codice, arrivano i suggerimenti per completare il codice che abbiamo nell'editor, bisogna valutare se raggiungono o meno il fine che avevamo in mente, a volte sono straordinariamente accurati, altre volte meno, e poi si fanno domande in chat su come ottenere risultati specifici, si chiede di scrivere i test, e così via. Qui è una specie di "maieutica" dove cerchiamo come Socrate di arrivare alla conoscenza che è sempre stata dentro il modello, o più semplicemente al risultato sperato. Chissà cosa ne penserebbe Platone, chissà se scriverebbe uno dei suoi dialoghi con un Large Language Model tra i protagonisti affianco a un Crizia e a un Timeo. Eppure, porre le domande giuste non è mai stato così importante per arrivare al nostro obiettivo. Altri editor come Cursor.ai, tentano un approccio ancora più innovativo, ripensando in pieno la creatività degli sviluppatori a cominciare dal

loro ambiente di sviluppo e dagli automatismi dell'attività di sviluppo software.
Esistono già diverse IA che permettono di generare codice, come interi siti web, a partire da un disegno sul tovagliolo. Quando fu mostrato dall'allora presidente di OpenAI Greg Brockman alla presentazione[15] di GPT4 fu una applicazione sbalorditiva, oggi invece è alla portata di quasi tutti i modelli avanzati. I progressi sono infatti continui, inesorabili. Github Copilot, uno strumento che ho iniziato a usare appena dopo il lancio nel 2021, forniva inizialmente suggerimenti di completamento codice a malapena utilizzabili e molto lenti. Quando adesso converso con il modello e da Visual Studio Code e premo il tasto "inserisci nell'editor" le modifiche appaiono direttamente nel mio codice e sembrano qualcosa di magico. Servono ancora gli sviluppatori, ma più che altro servono persone con una visione architetturale della soluzione, che sappiano dove vogliono arrivare e il valore di quello che stanno cercando di realizzare. Sapere quello che si vuole ottenere è diventato più importante di sapere come farlo. Se si vede la roadmap di Github, presentata al recente evento Github Universe[16], non siamo molto lontani da un mondo in cui sarà possibile chiedere "fammi un social network come Facebook, ma per i gatti", oppure "fammi un applicazione che gestisca l'inventario e la fatturazione della mia cartoleria" e ottenere un risultato all'altezza delle aspettative. Esistono già applicazioni IA di design come Uizard[17] che permettono di creare qualcosa di elegante e modificabile al volo. Sono pochi gli esperti della materia che sembrano al riparo da radicali sconvolgimenti.
Ma la vera domanda è: cosa impedirà a questi fornitori dei modelli di intelligenza artificiale di accentrarsi e dominare il mercato come è avvenuto in passato? Cosa impedirà l'emergere di nuovi monopoli simili a quelli che già vediamo oggi giorno? La buona, anzi, l'ottima notizia, è che questi modelli sono addestrati con algoritmi più o meno noti alla comunità degli esperti. Due aziende che addestrano il proprio modello con gli stessi algoritmi e gli stessi dati otterranno un risultato molto simile.
A riprova di questo, è notizia recente che Github Copilot, ovvero l'applicazione più popolare della Generative AI al campo dello sviluppo software, d'ora in poi supporterà non solo modelli di OpenAI come già faceva (GPT3.5 per il completamento codice, GPT4o per le conversazioni), ma anche quelli di Anthropic, Google e i modelli open source di Meta (Llama). Sicuramente sono ottime alternative, Claude Sonnet è già ritenuto un modello eccellente per questo tipo di applicazioni. Badate questo non è un dettaglio tecnico, ma un dato importantissimo. Github è al 100% proprietà di Microsoft, e Microsoft è quell'azienda che ha investito incredibili quantità di denaro in OpenAI, decine di miliardi di dollari, oltre ad avere fornito la capacità computazionale per addestrare i modelli, e reso disponibili quelli stessi modelli ai clienti del proprio cloud Azure. Inoltre, Github Copilot è attualmente forse l'unica applicazione di Generative AI che sta generando soldi veri, avendo un flusso di denaro di oltre due miliardi di dollari. Bisogna mettere questa notizia in prospettiva con quel rapporto interno[18] di Google "trapelato" (o fatto trapelare al pubblico, la differenza è sottile). Diceva «we have no moat, and neither does OpenAI», «non abbiamo un recinto protettivo, e non ce l'ha nemmeno OpenAI». L'open source e i modelli non proprietari di IA finiranno inevitabilmente per raggiungerci, concludeva il memorandum. Google e OpenAI sono i più grandi rivali nel settore, avendo praticamente effettuato tutta la ricerca di base che ha consentito i Large Language Models. Non ci sono per ora "recinti protettivi" a difesa di possibili monopoli nel campo della Generative AI. Non ci sono grandi segreti nascosti che offrano vantaggi competitivi duraturi, o che permettano di stabilire un monopolio duraturo. Gli algoritmi e le procedure sono noti, due addestramenti del modello sugli stessi dati e con la stessa potenza computazionale, otterranno risultati simili. Anche Microsoft lo ha riconosciuto in uno dei suoi prodotti di punta. Magari Claude è leggermente migliore, un po' più efficiente, ottimizzato meglio, quell'altro modello invece svolge un altro compito meglio, ma alla fine se non è zuppa, è pan bagnato, almeno nelle categorie di modello. Ogni giorno un modello ne scavalca un altro nei benchmark come Chatbot Arena[19] o in

quelle open source come LLM Arena di Hugging Face[20]. Questa per noi utenti finali è un'eccellente notizia. Probabilmente potremmo utilizzare modelli open source o addirittura addestrati da noi[21] per le nostre applicazioni e questa tecnologia potrebbe rivelarsi estremamente più democratica di quanto avessimo inizialmente temuto. Di questo dobbiamo ringraziare chi ha creduto a dei modelli open source, quindi grazie Hugging Face, grazie Meta (Llama), grazie Yann LeCun e grazie a molti altri per averci permesso di sognare un futuro più libero, democratico e aperto. Potremmo usare i modelli open source per scrivere il nostro codice. Inoltre, c'è un sacco di spazio "là sotto", nella scala del piccolo, non solo in quella del grande. Non bisogna guardare solo ai modelli più potenti e generici, ma a quelli più efficienti, leggeri e specializzati. Già esistono app come PocketPal e Apollo.ai che girano sul telefonino e scaricano un modello di LLM da 2 GB sul nostro cellulare. Si, avete capito bene: un chatbot capace di rispondere a quasi tutte le nostre domande sul vostro cellulare, non su un remoto server in mano a oscure entità, ma sul vostro dispositivo. Per questo sono sempre stato un grande fan di Mistral AI, l'azienda francese pioniera di questi modelli così compatti. Il loro modello iniziale 7b rilasciato a settembre 2023 occupava sulla mia macchina solo 80 MB di memoria, un quarto del mio client di posta elettronica. Mi ha fatto capire che il futuro è tutt'altro che scritto, e forse molto meno cupo di quanto avessimo inizialmente pensato. Questi Large Language Models open source, disponibili a tutti, saranno molto abili nello scrivere software e offriranno a tutti strumenti formidabili. Sull'intelligenza artificiale si può essere ottimisti[22] o scettici[23] - o forse "realisti" è la parola giusta -, ma comunque la si giri questa è una eccellente notizia.

Potremmo presto scrivere tutti rapidamente il software di cui abbiamo bisogno, senza usare strumenti costosi e proprietari. Tra dieci anni, avremmo ancora la lista dei dieci uomini (e speriamo anche donne, stavolta) più ricchi del pianeta, e saranno sempre ricchi oltre ogni immaginazione. È però improbabile che tra dieci anni abbiano ancora a che fare così tanto con il software, visto che il software sarà finalmente davvero alla portata di tutti.

P.S. Una domanda fondamentale: possiamo usare gli LLM per scrivere i programmi tradizionali che ci servono, ma possiamo usarlo per addestrare l'intelligenza artificiale? La risposta breve è: solo fino a un certo punto, perché quest'addestramento deriva dai dati. La risposta lunga richiederebbe un ulteriore articolo, e forse molto di più.

[1] Purtroppo, nella lista non figura attualmente nessuna donna: Wikipedia, Persone più ricche del mondo secondo Forbes.
[2] M. Andreessen, Why Software Is Eating the World, «Andreessen Horowitz (a16z)», 20 agosto 2011.
[3] Wikipedia, Guasto informatico di CrowdStrike del 2024.
[4] R. Piccolo, Treni, tutti gli errori che li hanno bloccati a Roma, «Wired», 3 ottobre 2024.
[5] Redazione, Un ospedale su tre subisce cyberattacchi. E il fenomeno è in aumento. L'allarme Onu e Oms, «Quotidiano Sanità», 11 novembre 2024.
[6] Redazione Economia e finanza, Bancomat, problemi col Pos in Italia per lavori alla rete del gas in Svizzera, «Rai News», 29 novembre 2024.
[7] Il Bitcoin oggi viene scambiato a 100.000 dollari. La similitudine con l'oro del Bitcoin non può però celare una verità: l'oro esisterebbe finché non distrutto a livello atomico, il Bitcoin senza l'operatività di una robusta rete di calcolatori non esisterebbe affatto.
[8] C. Accoto, Il mondo dato. Cinque brevi lezioni di filosofia digitale, Egea, Milano 2017.
[9] I. Jacobson, G. Booch e J. Rumbaugh, The Rational Unified Process, Addison-Wesley, Boston 1999.

[10] S. McConnell, Software Estimation. Demystifying the Black Art, Addison-Wesley, 2006.
[11] Jason Wei et al., Emergent Abilities of Large Language Models, «arXiv», (2206.07682v2), 26 ottobre 2022.
[12] R. Manzotti, E se l'intelligenza artificiale sentisse qualcosa?, «arXiv», 11 dicembre 2024.
[13] D.J. Chalmers, Could a Large Language Model be Conscious?, «arXiv», (2303.07103v3), 28 novembre 2022.
[14] Probabilmente cambierà la prospettiva richiesta agli sviluppatori, dallo scrivere singole parti di codice, al comprendere come le parti insieme soddisfano i requisiti dentro il sistema.
[15] M.R. Das, Code Comfort: Man draws website idea on a napkin, shows GPT-4, AI bot codes it in seconds, «Firstpost», 15 marzo 2023.
[16] https://githubuniverse.com
[17] https://uizard.io
[18] D. Patel e A. Ahmad, Google "We Have No Moat, And Neither Does OpenAI" Leaked Internal Google Document Claims Open Source AI Will Outcompete Google and OpenAI, «SemiAnalysis», 4 maggio 2023.
[19] https://lmarena.ai
[20] https://huggingface.co/open-llm-leaderboard
[21] Ci sono concrete speranze che l'addestramento dei modelli possa diventare molto più efficiente di oggi tramite nuovi approcci.
[22] E. Mollick, Co-Intelligence. Living and Working with AI, Portfolio, 2024.
[23] A. Narayan e S. Kapoor, AI Snake Oil: What Artificial Intelligence Can Do, What It Can't, and How to Tell the Difference, Princeton University Press, Princeton 2024.