Más allá del hype de la inteligencia artificial Por qué los Modelos de Lenguaje Grandes transformarán el software, pero no lo sustituirán

Versión 1.0 (edición de publicación)

Antonio Latorre

Resumen

Modelos de Lenguaje Grandes (LLM) ha transformado el desarrollo de software, el trabajo del conocimiento y la interacción entre humanos y computadoras al permitir a los usuarios expresar sus intenciones en lenguaje natural. Este artículo sostiene que la afirmación dominante de que la IA reemplazará al software empresarial se basa en un error de categoría: sistemas de Modelos de Lenguaje Grandes (LLM) optimizados para la generación de lenguaje plausible, mientras que se requiere que el software empresarial proporcione ejecución determinista, reproducibilidad, gobernanza, auditabilidad, seguridad y computación matemáticamente confiable. El artículo hace tres contribuciones originales. Primero, propone una teoría de tres generaciones de informática: informática determinista, informática probabilística y Computación Orientada por Intención (IDC). En segundo lugar, define Espejismo de Competencia de la IA y Brecha de Confianza Computacional como conceptos explicativos del desajuste entre la competencia lingüística percibida y las garantías computacionales. En tercer lugar, propone el modelo arquitectónico para sistemas empresariales de próxima generación: un Capa de Interfaz Probabilística (PIL) que interpreta la intención humana, un Capa de Inteligencia de Dominio (DIL) que encapsula el conocimiento del dominio y la metodología empresarial, y un Capa de Ejecución Determinista (DEL) que realiza cálculos confiables. Económicamente, el artículo sostiene que la IA no elimina el software empresarial; en cambio, reduce el costo de creación de software al tiempo que aumenta la importancia estratégica de la inteligencia de dominio, la arquitectura, la gobernanza y la confianza operativa.

Palabras clave: Modelos de Lenguaje Grandes (LLM); Computación Orientada por Intención (IDC); arquitectura empresarial; Capa de Interfaz Probabilística (PIL); Capa de Inteligencia de Dominio (DIL); Capa de Ejecución Determinista (DEL); Brecha de Confianza Computacional; Espejismo de Competencia de la IA.

Tabla de contenidos

1. Introducción

La aparición de Modelos de Lenguaje Grandes (LLM) representa uno de los avances tecnológicos más significativos en la historia de la informática. En sólo unos pocos años, sistemas como ChatGPT, Claude,

Gemini y otros han demostrado una capacidad sin precedentes para comprender el lenguaje natural, generar software, resumir información compleja, ayudar en la investigación científica e interactuar con los humanos en formas que, hasta hace poco, parecían inalcanzables.

La velocidad de este progreso tecnológico ha generado un enorme entusiasmo en prácticamente todas las industrias. Al mismo tiempo, también ha creado un nivel igualmente sin precedentes de malentendidos sobre el papel real de la Inteligencia Artificial dentro de los sistemas de software modernos.

Un número creciente de artículos, presentaciones en conferencias y debates públicos sostienen que el software empresarial pronto quedará obsoleto porque los usuarios simplemente "pedirán a una IA" que realice cualquier tarea que requieran. Según esta narrativa, las interfaces conversacionales reemplazarán gradualmente a las aplicaciones tradicionales, reduciendo drásticamente la necesidad de plataformas de software especializadas.

Este artículo sostiene que tales conclusiones surgen de una confusión fundamental entre dos paradigmas computacionales que, aunque estrechamente relacionados, resuelven problemas fundamentalmente diferentes.

Los sistemas de software tradicionales son motores computacionales deterministas. Dados los mismos insumos y el mismo estado interno, se espera que produzcan exactamente los mismos resultados, siempre. Este determinismo constituye la base sobre la que se han construido durante décadas los sistemas financieros, los dispositivos médicos, las redes de telecomunicaciones, la automatización industrial, los sistemas de transporte y prácticamente todas las aplicaciones empresariales de misión crítica.

Los modelos de lenguaje grandes, por el contrario, pertenecen a una categoría computacional fundamentalmente diferente. Son modelos probabilísticos entrenados para estimar la continuación más probable de una secuencia de tokens dado un contexto particular. Sus notables capacidades surgen no de la ejecución de algoritmos predefinidos que garantizan resultados matemáticamente correctos, sino del aprendizaje de representaciones estadísticas del lenguaje y el conocimiento a partir de enormes cantidades de datos.

Esta distinción es considerablemente más importante de lo que pueda parecer inicialmente.

Muchos debates públicos en torno a la Inteligencia Artificial suponen implícitamente que, dado que los LLM a menudo producen respuestas correctas, coherentes y altamente sofisticadas, operan como motores de razonamiento determinista capaces de reemplazar el software tradicional. Si bien esta suposición es comprensible, no refleja con precisión los principios computacionales sobre los que se basan estos sistemas.

En realidad, Modelos de Lenguaje Grandes (LLM) y el software determinista exhiben fortalezas complementarias.

Los LLM se destacan en la interpretación de intenciones humanas ambiguas, la comunicación en lenguaje natural, la generación de código, la síntesis de conocimientos y la asistencia en tareas cognitivas complejas. El software determinista sobresale en la ejecución de algoritmos definidos formalmente, haciendo cumplir reglas de negocio, garantizando la reproducibilidad, manteniendo la coherencia transaccional, satisfaciendo los requisitos regulatorios y produciendo resultados matemáticamente verificables __KEEP[40], [32], [36]-[40].

En consecuencia, este artículo propone que el futuro de la informática empresarial no debe entenderse como una transición del software a la Inteligencia Artificial.

Más bien, representa el surgimiento de una nueva arquitectura computacional en la que la inteligencia probabilística se convierte en la interfaz principal entre los humanos y los sistemas computacionales deterministas.

Dentro de esta arquitectura, la Inteligencia Artificial es responsable de comprender la intención, mientras que el software determinista sigue siendo responsable de ejecutar la lógica empresarial, el cálculo matemático, la optimización, la gobernanza, la auditoría, la seguridad y la reproducibilidad [30]-[30].

Esta separación arquitectónica permite a las organizaciones beneficiarse simultáneamente de la flexibilidad de la interacción del lenguaje natural y la confiabilidad requerida por los sistemas empresariales de misión crítica.

Las implicaciones de esta distinción se extienden mucho más allá de la ingeniería de software.

Remodelan fundamentalmente la forma en que las organizaciones deberían pensar sobre las aplicaciones empresariales, el desarrollo de software, la interacción persona-computadora, la gobernanza, la regulación y la ventaja competitiva en la era de la Inteligencia Artificial.

Por lo tanto, la tesis central de este artículo se puede resumir de la siguiente manera:

Los modelos de lenguaje grandes no representan el fin del software determinista. Representan el surgimiento de un nuevo paradigma computacional en el que la inteligencia probabilística orquesta la ejecución determinista.

Esta tesis desafía la percepción cada vez más común de que la IA conversacional eliminará la necesidad de software empresarial. Más bien, sugiere que el software está entrando en una nueva etapa evolutiva: una en la que el lenguaje se convierte en la interfaz universal, mientras que la computación determinista sigue siendo la base de la que siguen dependiendo la confianza, la gobernanza y la ejecución empresarial.

2. Tres generaciones de informática

A lo largo de su historia, la informática ha evolucionado reduciendo progresivamente el esfuerzo cognitivo necesario para que los humanos interactúen con las máquinas. Cada revolución tecnológica importante no ha reemplazado a la generación anterior, sino que ha introducido una nueva capa de abstracción que permitió formas cada vez más naturales de comunicación entre humanos y computadoras.

Desde esta perspectiva, la aparición de los Grandes Modelos de Lenguaje no debe entenderse simplemente como un avance más de la Inteligencia Artificial. Más bien, representa el comienzo de un tercer paradigma computacional, que cambia fundamentalmente el papel del software dentro de los sistemas empresariales.

Proponemos que la computación moderna puede entenderse como la evolución de tres generaciones distintas: Computación determinista, Computación probabilística y Computación impulsada por la intención.

Aunque estas generaciones coexisten hoy, cada una introduce una relación fundamentalmente diferente entre los humanos, el software y la computación.

Primera generación: computación determinista

La primera generación abarca prácticamente todo el software tradicional desarrollado desde el nacimiento de la informática digital.

Su característica definitoria es el determinismo.

Dadas entradas idénticas y estados del sistema idénticos, se espera que el software produzca resultados idénticos cada vez que se ejecuta. La previsibilidad, la repetibilidad y la corrección matemática constituyen los fundamentos de este paradigma.

Dentro de la informática determinista, todos los comportamientos posibles deben ser diseñados e implementados explícitamente por ingenieros de software. Las reglas comerciales están codificadas como algoritmos, los flujos de trabajo están predefinidos y cada interacción del usuario sigue una secuencia anticipada durante el desarrollo.

La comunicación entre humanos y máquinas se produce a través de interfaces rígidas, que incluyen instrucciones de línea de comandos, menús gráficos, formularios, botones, API y flujos de trabajo predefinidos.

En este paradigma, los desarrolladores de software actúan como traductores entre las intenciones humanas y la ejecución de las máquinas. Cada nuevo requisito empresarial generalmente requiere el desarrollo de un nuevo software.

Este enfoque ha permitido la construcción de prácticamente todos los sistemas digitales críticos de la sociedad moderna, incluidas plataformas financieras, redes de telecomunicaciones, dispositivos médicos,

sistemas de planificación de recursos empresariales, automatización industrial, sistemas de transporte e infraestructuras informáticas científicas.

A pesar de su enorme éxito, la computación determinista presenta una limitación importante.

Los humanos deben adaptar su comunicación al lenguaje del software en lugar de que el software se adapte al lenguaje de los humanos.

Segunda Generación: Computación Probabilística

La aparición de los grandes modelos de lenguaje introdujo una capacidad computacional fundamentalmente diferente.

En lugar de requerir que los humanos se comunicaran a través de interfaces predefinidas, las computadoras se volvieron capaces de interpretar el lenguaje natural directamente.

Esto representa un cambio profundo.

Por primera vez, los usuarios de software ya no necesitan comprender menús, API, lenguajes de programación o flujos de trabajo complejos para realizar tareas sofisticadas.

En cambio, simplemente describen sus intenciones utilizando el lenguaje humano común.

Esta capacidad está habilitada por redes neuronales probabilísticas entrenadas para estimar la probabilidad condicional de secuencias lingüísticas [13], [9]-[13].

A diferencia del software determinista, estos sistemas no ejecutan procedimientos de razonamiento programados explícitamente.

Más bien, generan respuestas prediciendo la continuación más probable de una secuencia de tokens dado el contexto circundante.

Como consecuencia, sus resultados son inherentemente probabilísticos.

Para preguntas muy restringidas, la distribución de probabilidad se vuelve extremadamente concentrada, lo que hace que las respuestas parezcan deterministas.

Sin embargo, para los problemas abiertos, surgen naturalmente múltiples respuestas válidas porque numerosas continuaciones pueden exhibir probabilidades igualmente altas.

Esta naturaleza probabilística proporciona una flexibilidad extraordinaria.

Permite que el sistema interprete instrucciones ambiguas, genere software, resuma documentos, traduzca idiomas, explique conceptos y ayude en el trabajo creativo.

Sin embargo, también introduce una limitación importante.

Debido a que los resultados se generan de forma probabilística y no algorítmica, la corrección no puede garantizarse únicamente mediante el modelo de lenguaje en sí.

El modelo optimiza la plausibilidad en lugar de la corrección formal [18], [16]-[18].

En consecuencia, la computación probabilística expande dramáticamente la interacción persona-computadora y al mismo tiempo requiere nuevos mecanismos para garantizar la confiabilidad siempre que los resultados deterministas sean esenciales.

Tercera generación: informática basada en la intención

Sostenemos que la próxima etapa de la informática no es la sustitución del software determinista por la Inteligencia Artificial probabilística.

Más bien, es la integración de ambos paradigmas en una arquitectura computacional unificada.

Nos referimos a este paradigma emergente como Computación Orientada por Intención (IDC).

> Definición 1: Computación Orientada por Intención (IDC). Computación basada en intenciónComputación Orientada por Intención (IDC)paradigma de computación en el que los objetivos humanos se expresan principalmente como intención de lenguaje natural, interpretados por sistemas de lenguaje probabilístico, restringidos por inteligencia de dominio explícita y ejecutados a través de mecanismos computacionales deterministas que siguen siendo auditables, reproducibles y gobernables.

Dentro de la Computación basada en la intención, el lenguaje natural se convierte en la interfaz principal a través de la cual los humanos expresan objetivos, limitaciones y preferencias.

Los Grandes Modelos de Lenguaje son responsables de interpretar estas intenciones, resolver ambigüedades lingüísticas y transformar las solicitudes humanas en instrucciones computacionales estructuradas.

Sin embargo, la ejecución sigue siendo responsabilidad de los sistemas de software deterministas.

Las reglas comerciales, los algoritmos de optimización, el cálculo numérico, la coherencia transaccional, la gobernanza, la auditoría, la seguridad y el cumplimiento normativo continúan siendo implementados mediante motores de ejecución deterministas cuyo comportamiento es matemáticamente reproducible.

Esta separación cambia fundamentalmente el papel de la Inteligencia Artificial.

En lugar de reemplazar el software, la IA se convierte en una capa de orquestación ubicada entre la cognición humana y la computación determinista.

Su propósito ya no es ejecutar procesos de negocio directamente, sino traducir las intenciones humanas en flujos de trabajo computacionales deterministas.

Esta arquitectura combina las principales fortalezas de ambos paradigmas al tiempo que agrega un Capa de Inteligencia de Dominio (DIL) dedicado para el conocimiento, la metodología y la gobernanza empresarial.

La inteligencia probabilística aporta flexibilidad, interacción natural y comprensión contextual.

El software determinista aporta reproducibilidad, confiabilidad, explicabilidad, gobernanza y certeza computacional [30]-[30].

Juntas, estas capacidades complementarias permiten sistemas empresariales que son simultáneamente intuitivos para los humanos y confiables para las organizaciones.

La evolución de la interacción persona-computadora

Vistas desde una perspectiva histórica, estas tres generaciones revelan una dirección común.

Cada generación reduce la cantidad de conocimientos técnicos necesarios para que los humanos interactúen con las computadoras.

La progresión se puede resumir de la siguiente manera:

Como se muestra en la Tabla 1, esta progresión pasa de la interacción estructurada hombre-máquina hacia Computación Orientada por Intención (IDC).

Tabla 1. Tres Generaciones de Computación. La tabla resume la evolución de la interacción determinista a la generación de lenguaje probabilístico y Computación Orientada por Intención (IDC).

Generación	Comunicación humana	Ejecución de la máquina	Característica primaria
Computación determinista	Comandos, formularios, menús, API	Algoritmos deterministas	Previsibilidad
Computación probabilística	lenguaje natural	Generación de lenguaje probabilístico.	Flexibilidad
Computación impulsada por la intención	Intenciones del lenguaje natural	Ejecución determinista orquestada por IA	Autonomía confiable

Esta progresión sugiere que la Inteligencia Artificial no debe verse como un sustituto de la ingeniería de software.

Más bien, representa la evolución de la interfaz humana a través de la cual se instruyen los sistemas de software deterministas.

En consecuencia, es poco probable que el futuro de la informática empresarial consista en que los agentes conversacionales reemplacen las aplicaciones empresariales.

Más bien, las propias aplicaciones empresariales incorporarán cada vez más inteligencia probabilística como su interfaz principal, preservando al mismo tiempo los núcleos computacionales deterministas responsables de la ejecución, la gobernanza y la confianza.

La Figura 4 proporciona una descripción visual del argumento de los resultados empresariales confiables desarrollado en esta sección.

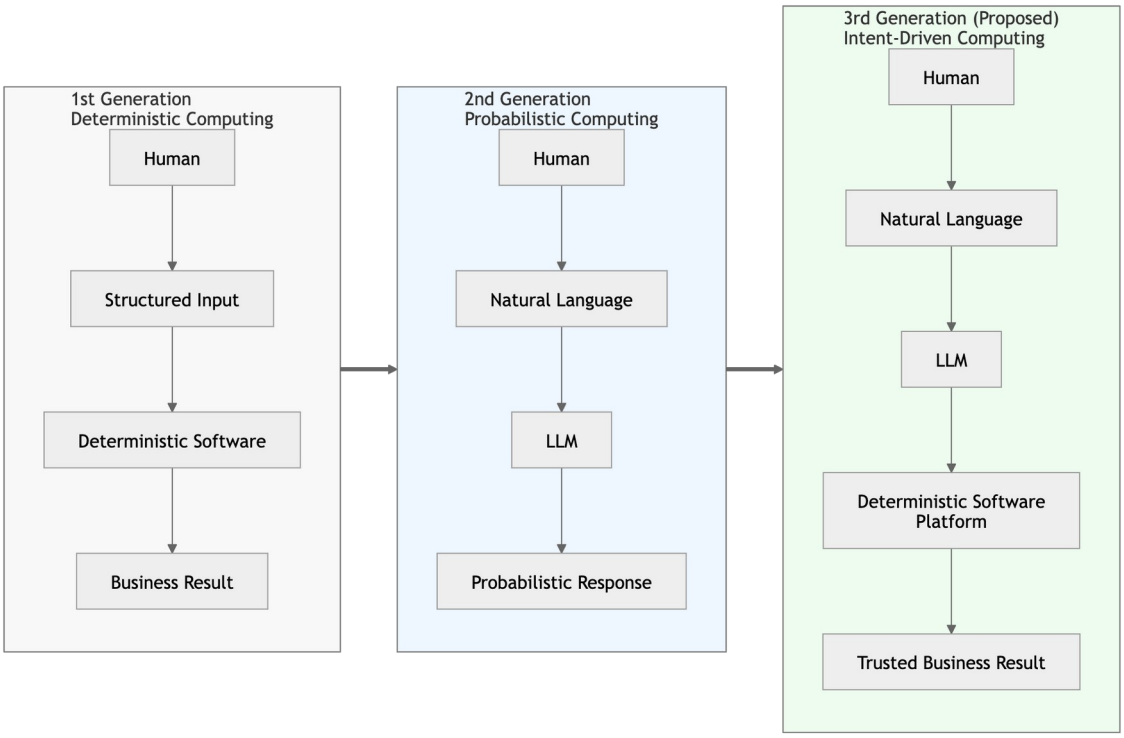

Figura 4. Intuición arquitectónica para resultados empresariales confiables. La arquitectura actual formalizada en este documento es PIL -> DIL -> DEL; la figura preserva la intuición visual de que la intención mediada por la IA debe, en última instancia, conectarse con la ejecución gobernada.

3. Cómo funcionan los modelos de lenguaje grandes

Comprender la naturaleza computacional de los modelos de lenguajes grandes es esencial para comprender tanto sus extraordinarias capacidades como sus limitaciones inherentes.

Gran parte del debate público en torno a la Inteligencia Artificial trata a los LLM como si fueran motores de razonamiento capaces de calcular verdades objetivas sobre el mundo. Si bien sus respuestas frecuentemente exhiben una coherencia y sofisticación notables, el mecanismo computacional subyacente es fundamentalmente diferente del software determinista tradicional.

En esencia, los modelos de lenguaje grande son redes neuronales probabilísticas entrenadas para estimar la distribución de probabilidad condicional del siguiente token dados todos los tokens observados previamente [13], [9]-[13].

Formalmente, dada una secuencia de fichas

$$x_1, x_2, \ldots, x_n$$

el modelo intenta estimar

$$P\left(x_{n+1}, / , x_1, x_2 \ldots x_n\right)$$

En otras palabras, el modelo responde continuamente a una única pregunta matemática:

Teniendo en cuenta todo lo que se ha escrito hasta ahora, ¿cuál es la siguiente señal más probable? [13]_, [10]-[13]

Cada palabra, signo de puntuación y símbolo generado se produce repitiendo este proceso de predicción miles de veces hasta que se haya generado la respuesta completa.

Es importante destacar que el modelo no recupera respuestas predefinidas de una base de datos ni ejecuta procedimientos de razonamiento simbólico similares a los implementados en los sistemas de software clásicos.

En cambio, cada predicción resulta de la evaluación de miles de millones de parámetros aprendidos que codifican relaciones estadísticas adquiridas durante el entrenamiento.

Las notables capacidades de los LLM modernos surgen de la escala de estas representaciones aprendidas más que de reglas de razonamiento explícitamente programadas.

El modelado del lenguaje es un proceso probabilístico

Para cada posición en la respuesta generada, el modelo calcula una distribución de probabilidad sobre cada token de su vocabulario.

Conceptualmente, supongamos que el usuario pregunta:

¿Cuál es la capital de Francia?

Inmediatamente antes de generar la palabra final, el modelo evalúa internamente muchas posibles continuaciones.

A modo de ejemplo, la distribución podría parecerse a:

Como se muestra en la Tabla 2, un caso de baja entropía puede hacer que un modelo probabilístico parezca determinista porque una continuación domina la distribución.

Tabla 2. Distribución de tokens de baja entropía. El ejemplo ilustra por qué indicaciones fácticas simples pueden parecer deterministas incluso cuando las genera un modelo probabilístico.

Ficha de candidato	Probabilidad estimada
París	99,9999%
León	0,00008%
Marsella	0,00002%
Londres	<0,00001%

Estas cifras son puramente ilustrativas.

La observación importante no son las probabilidades específicas, sino la existencia de una distribución de probabilidad en sí misma.

El modelo, por su objetivo básico de modelado del lenguaje, no verifica de forma independiente si "París" es objetivamente cierto.

En cambio, estima que "París" es abrumadoramente más probable que cualquier alternativa dado el contexto de la conversación y la estructura estadística aprendida durante el entrenamiento.

Esta distinción puede parecer sutil, pero cambia fundamentalmente cómo se deben interpretar los sistemas de IA.

Predicción en lugar de verificación

El software tradicional frecuentemente calcula los resultados ejecutando algoritmos especificados formalmente.

Un sistema bancario que calcula el interés compuesto aplica ecuaciones matemáticas.

Una base de datos que ejecuta una consulta SQL sigue planes de ejecución deterministas.

Un algoritmo de clasificación produce exactamente el mismo orden cada vez que recibe entradas idénticas.

Los modelos de lenguaje grandes funcionan de manera diferente.

Generan resultados prediciendo continuaciones lingüísticas plausibles en lugar de verificar proposiciones con modelos matemáticos formales.

En consecuencia, la exactitud de una respuesta generada es una consecuencia emergente del aprendizaje estadístico exitoso, no una garantía matemática proporcionada por el proceso computacional en sí.

Esta distinción explica una de las características más incomprendidas de la IA moderna.

Un modelo de lenguaje grande puede generar una respuesta completamente correcta.

También puede generar una respuesta parcialmente correcta.

O uno que sea completamente incorrecto sin dejar de ser gramaticalmente coherente y muy convincente.

El mecanismo computacional sigue siendo exactamente el mismo en los tres casos.

Por qué las preguntas sencillas suelen producir respuestas estables

Inmediatamente surge una objeción común.

Si los LLM son probabilísticos, ¿por qué casi siempre responden:

París

cuando se le preguntó sobre la capital de Francia?

La respuesta está en la forma de la distribución de probabilidad subyacente.

Para preguntas fácticas muy restringidas, un token candidato a menudo recibe una probabilidad abrumadoramente mayor que cualquier alternativa.

Aunque el proceso de generación sigue siendo probabilístico, la distribución está tan concentrada que las ejecuciones repetidas frecuentemente producen resultados idénticos.

Este aparente determinismo no debe confundirse con el verdadero cálculo determinista.

El modelo sigue siendo probabilístico.

La distribución de probabilidad simplemente muestra una entropía extremadamente baja.

A medida que las indicaciones se vuelven más amplias, más creativas o menos limitadas, la masa de probabilidad se extiende naturalmente a través de muchas posibles continuaciones.

En consecuencia, múltiples respuestas diferentes se vuelven igualmente plausibles.

El siguiente mensaje ilustra este fenómeno:

Diseñe un canal de aprendizaje automático para la predicción del riesgo crediticio.

No existe una continuación única.

Diferentes ejecuciones pueden recomendar legítimamente diferentes algoritmos, estrategias de validación, técnicas de ingeniería de características, procedimientos de optimización de hiperparámetros o arquitecturas de implementación.

Cada respuesta puede ser técnicamente válida.

Ninguno está determinado únicamente por el mensaje.

Las consecuencias para los sistemas empresariales

Este comportamiento probabilístico constituye una de las mayores fortalezas de los modelos de lenguaje grande.

Permite flexibilidad, creatividad, comprensión contextual e interacción del lenguaje natural que el software determinista no puede lograr fácilmente.

Sin embargo, la misma característica también limita su capacidad para servir como único motor computacional para sistemas empresariales que requieren:

reproducibilidad, [30]-[30]

- revisión de cuentas,
- gobernancia,
- cumplimiento normativo,
- coherencia transaccional,
- corrección matemática,
- ejecución determinista.

Las organizaciones que operan sistemas críticos no pueden depender exclusivamente de procesos computacionales cuyos resultados pueden variar legítimamente entre ejecuciones.

En consecuencia, los LLM deben entenderse no como reemplazos de la computación determinista, sino como intérpretes probabilísticos capaces de traducir las intenciones humanas en flujos de trabajo computacionales ejecutados posteriormente por software determinista.

Esta distinción arquitectónica forma la base conceptual de la próxima generación de informática empresarial propuesta a lo largo de este documento.

Los modelos de lenguaje grandes no deciden: toman muestras

Uno de los conceptos erróneos más extendidos sobre la Inteligencia Artificial moderna es la creencia de que los Grandes Modelos de Lenguaje deciden qué responder.

Desde una perspectiva computacional, esto no es lo que realmente sucede.

Un LLM no concluye internamente que una respuesta es objetivamente correcta y rechaza todas las alternativas.

En cambio, después de procesar el contexto de entrada, el modelo produce una distribución de probabilidad sobre cada posible siguiente token en su vocabulario [13]_, [10]-[13].

Conceptualmente, el modelo puede estimar algo similar a:

Como se muestra en la Tabla 3, la probabilidad simbólica explica la plausibilidad pero no constituye en sí misma una garantía de verdad formal [18], [16]-[18].

Tabla 3. Probabilidades de tokens candidatos. La distribución ilustrativa muestra cómo la plausibilidad puede concentrarse en torno a una respuesta correcta sin convertirse en una garantía formal de verdad.

Candidato	Probabilidad estimada
París	99,9999%
León	0,00008%
Marsella	0,00002%
Londres	<0,00001%

El modelo en sí se detiene ahí.

No "elige París".

Simplemente calcula que París tiene la probabilidad condicional más alta dado el contexto [13], [9]-[13].

Luego, un procedimiento de decodificación separado es responsable de producir el siguiente token [13]_, [10]-[13].

Dependiendo de la estrategia de decodificación, el sistema puede:

- seleccione siempre el token de mayor probabilidad (decodificación codiciosa);
- buscar en múltiples secuencias candidatas (búsqueda de haz);
- muestrear aleatoriamente a partir de la distribución de probabilidad (muestreo);
- restringir el muestreo a los candidatos más probables (top-k);

- muestrear solo de la masa de probabilidad más pequeña que exceda un umbral (muestreo top-p/núcleo);
- ajustar la diversidad a través de un parámetro de temperatura.

Estas estrategias de decodificación no cambian el conocimiento subyacente aprendido por el modelo.

Simplemente determinan cómo se convierte la distribución de probabilidad en texto observable.

Esta distinción es esencial porque revela que la aparente certeza de una respuesta LLM no debe confundirse con el cálculo determinista.

Incluso cuando dos ejecuciones producen resultados idénticos, el proceso computacional sigue siendo fundamentalmente probabilístico.

El modelo no ejecutó una prueba matemática.

Estimó probabilidades.

Por qué es importante el muestreo

Esta distinción tiene profundas implicaciones prácticas.

Supongamos que una organización le pregunta a un LLM:

Diseñar un canal de calificación crediticia.

No existe una única continuación matemáticamente correcta.

El modelo asigna probabilidad significativa a muchas alternativas técnicamente válidas.

Una ejecución puede recomendar LightGBM.

Otro puede recomendar CatBoost.

Un tercero puede priorizar la explicabilidad y proponer una regresión logística.

Las tres respuestas pueden ser técnicamente defendibles.

La modelo no cambia de opinión.

Está generando diferentes muestras a partir de un panorama de probabilidad que contiene múltiples continuaciones plausibles.

Esta flexibilidad constituye una de las mayores fortalezas de la IA moderna.

Permite la creatividad, la adaptación y el razonamiento contextual.

Sin embargo, también explica por qué los sistemas empresariales que requieren reproducibilidad no pueden delegar la ejecución final por completo a modelos probabilísticos [30]-[30].

Del muestreo a la ejecución determinista

La consecuencia no es que deban evitarse los LLM.

Todo lo contrario.

Su capacidad para interpretar intenciones humanas ambiguas no tiene precedentes.

El desafío radica en asignarles el rol computacional apropiado.

En entornos empresariales, el LLM debería funcionar principalmente como un intérprete de intenciones y no como el motor de ejecución final.

Una vez que se ha comprendido el objetivo del usuario, los componentes de software deterministas deben ejecutar los correspondientes algoritmos, reglas de negocio, procedimientos de optimización y mecanismos de gobernanza __KEEP[40], [32], [36]-[40].

En esta arquitectura, Capa de Interfaz Probabilística (PIL) determina lo que quiere el usuario, Capa de Inteligencia de Dominio (DIL) determina cómo se debe resolver el problema del dominio y Capa de Ejecución Determinista (DEL) ejecuta el resultado.

El software determinista ejecuta la tarea a través de procesos computacionales reproducibles, auditables y gobernados.

Esta separación combina las fortalezas de ambos paradigmas al tiempo que minimiza sus respectivas debilidades.

4. Verdad versus plausibilidad

Uno de los conceptos erróneos más importantes que rodean a los modelos de lenguajes grandes es la suposición de que están optimizados para descubrir la verdad. No lo son. Están optimizados para generar un lenguaje plausible.

Esta distinción está en el centro de prácticamente todos los malentendidos relacionados con la Inteligencia Artificial moderna.

Al interactuar con un LLM, los usuarios naturalmente evalúan la calidad de una respuesta de acuerdo con su exactitud fáctica.

Sin embargo, el modelo nunca fue entrenado con la corrección objetiva como objetivo principal de optimización.

En cambio, durante el preentrenamiento, el objetivo es considerablemente diferente.

El modelo ajusta continuamente miles de millones de parámetros internos para maximizar la probabilidad de predecir correctamente el siguiente token dado el contexto circundante [13]_, [10]-[13].

En otras palabras, el objetivo de optimización no es:

"Produzca la respuesta verdadera".

Es:

"Produzca la continuación que sea estadísticamente más consistente con todo lo observado hasta ahora".

Esta distinción es sutil pero profunda.

La verdad y la verosimilitud frecuentemente se superponen [18], [16]-[18].

No son el mismo concepto.

Cuando plausibilidad es igual a verdad

Para muchas preguntas, la continuación estadísticamente más probable también resulta objetivamente correcta.

Por ejemplo,

¿Cuál es la capital de Francia?

Debido a que "París" domina abrumadoramente la distribución estadística aprendida durante el entrenamiento, el modelo produce la respuesta correcta con una consistencia notable.

Similarmente,

¿A qué temperatura se congela el agua?

¿Quién escribió Hamlet?

¿Cuál es la raíz cuadrada de 81?

Estas preguntas suelen producir respuestas correctas porque el conocimiento humano muestra un acuerdo estadístico abrumador.

En estas situaciones, la plausibilidad se convierte en una excelente aproximación a la verdad [18], [16]-[18].

Esto explica por qué los LLM modernos parecen sorprendentemente precisos en muchos ámbitos.

Cuando la plausibilidad difiere de la verdad

La distinción se vuelve visible siempre que la plausibilidad estadística ya no identifica de manera única la corrección fáctica [18], [16]-[18].

Los ejemplos incluyen:

- preguntas ambiguas;
- información incompleta;
- eventos raros;
- situaciones novedosas;
- entidades fabricadas;
- fuentes contradictorias;
- dominios altamente especializados.

Considere el siguiente mensaje:

Describa las contribuciones del profesor John Smith a las finanzas cuánticas.

Si no existe tal investigador, el modelo frecuentemente produce una biografía convincente en lugar de reconocer la incertidumbre.

Es importante destacar que el modelo no fabrica información intencionalmente.

Está haciendo exactamente aquello para lo que fue entrenado.

Genera la continuación que parece más plausible estadísticamente dados los patrones lingüísticos que ha aprendido.

Desde la perspectiva del objetivo de optimización, este comportamiento es totalmente consistente.

Desde la perspectiva de la corrección fáctica, constituye una alucinación.

Por lo tanto, las alucinaciones no son defectos de software [18], [18].

Son consecuencias esperadas de optimizar la plausibilidad en lugar de la verdad [18], [16]-[18].

Por qué los humanos confunden verosimilitud con verdad

La cognición humana evolucionó en condiciones en las que el lenguaje fluido generalmente implicaba competencia [21], [19]-[21].

Durante la mayor parte de la historia de la humanidad, las explicaciones coherentes provinieron casi exclusivamente de otros humanos que poseían conocimientos relevantes.

Como consecuencia, nuestro cerebro asocia de forma natural características como:

- corrección gramatical;
- estructura lógica;
- confianza;
- vocabulario técnico;
- narrativa coherente.

con veracidad.

Los modelos de lenguaje grandes explotan exactamente estas regularidades estadísticas.

Sus resultados poseen todas las características lingüísticas que los humanos asocian instintivamente con la experiencia.

En consecuencia, los usuarios frecuentemente asignan un grado de credibilidad que excede lo que realmente garantiza el proceso computacional subyacente.

Este fenómeno psicológico contribuye significativamente a la percepción pública actual de que los LLM "saben" o "comprenden" los hechos de la misma manera que los expertos humanos.

En realidad, generan un lenguaje que se parece mucho a la comunicación experta.

Esos no son conceptos equivalentes.

Por qué es importante esta distinción

En aplicaciones cotidianas, maximizar la plausibilidad suele ser suficiente [18], [16]-[18].

Redacción de correos electrónicos.

Informes resumidos.

Generación de prototipos de software.

Lluvia de ideas.

Explicar conceptos.

Escritura creativa.

En estos ámbitos, naturalmente existen múltiples resultados aceptables.

Sin embargo, los sistemas empresariales frecuentemente operan bajo restricciones fundamentalmente diferentes.

Transacciones financieras.

Diagnóstico médico.

Cálculo de impuestos.

Aprobación de crédito.

Precios de seguros.

Informes regulatorios.

Automatización industrial.

En estos contextos, puede existir sólo un resultado computacional aceptable.

La plausibilidad es insuficiente.

La corrección determinista se vuelve obligatoria.

Esta distinción explica por qué los LLM no deberían reemplazar los motores computacionales deterministas en sistemas de misión crítica.

Más bien, deberían complementarlos.

La Inteligencia Artificial sobresale en interpretar la intención, reducir la ambigüedad y ayudar al razonamiento humano.

El software determinista se destaca por producir una ejecución confiable, reproducible y verificable.

Comprender dónde termina la plausibilidad y dónde se vuelve necesario el determinismo constituye uno de los principios arquitectónicos centrales propuestos a lo largo de este artículo [18], [16]-[18].

5. El Brecha de Confianza Computacional

Una de las consecuencias más importantes de los modelos de lenguajes grandes no es puramente computacional.

Es cognitivo.

Los LLM modernos generan un lenguaje con un extraordinario nivel de fluidez, coherencia y confianza. Sus respuestas se parecen mucho a las producidas por expertos humanos con mucho conocimiento. Como resultado, los usuarios naturalmente atribuyen un nivel de competencia y certeza que frecuentemente excede lo que el proceso computacional subyacente es realmente capaz de garantizar.

Nos referimos a este fenómeno psicológico como Espejismo de Competencia de la IA.

> Definición 2: Espejismo de Competencia de la IA. The IA CompeteEspejismo de Competencia de la IA: tendencia cognitiva de los usuarios humanos a inferir experiencia, veracidad o confiabilidad operativa a

> partir de un lenguaje fluido, coherente y seguro generado por IA, incluso cuando la salida del sistema se produce mediante generación probabilística y no va acompañada de una garantía computacional independiente.

El Espejismo de Competencia de la IA ocurre cuando los humanos confunden inconscientemente la competencia lingüística con la certeza computacional.

Debido a que los LLM se comunican utilizando un lenguaje que parece autoritario, internamente consistente y técnicamente sofisticado, los usuarios instintivamente infieren que el razonamiento subyacente también debe ser correcto.

En realidad, se trata de propiedades fundamentalmente diferentes.

Una respuesta puede ser lingüísticamente perfecta y al mismo tiempo ser objetivamente incorrecta, estadísticamente incierta o computacionalmente no verificable.

El lenguaje crea la apariencia de certeza.

Las matemáticas no.

Por qué los humanos caen en el espejismo

La cognición humana evolucionó en condiciones en las que el lenguaje fluido generalmente servía como un indicador confiable del conocimiento [21], [19]-[21].

A lo largo de la historia de la humanidad, las explicaciones coherentes casi siempre provinieron de personas que poseían una experiencia genuina.

En consecuencia, nuestros cerebros desarrollaron poderosas heurísticas cognitivas que asocian características como:

lenguaje fluido; [21], [19]-[21]

- vocabulario técnico;
- estructura coherente;
- progresión lógica;
- confianza;
- explicaciones persuasivas;

con competencia y veracidad.

Los modelos de lenguaje grandes reproducen precisamente estas características lingüísticas.

No engañan intencionalmente a los usuarios.

Más bien, generan un lenguaje cuyas propiedades estadísticas se parecen mucho a las que se encuentran en la comunicación humana experta.

Como consecuencia, los humanos asignan naturalmente un grado de credibilidad que a menudo excede las garantías matemáticas proporcionadas por el proceso computacional subyacente.

Esta reacción no es irracional ni accidental.

Es una consecuencia esperada de la interacción de la cognición humana con la generación probabilística del lenguaje.

Del Espejismo de Competencia de la IA al Brecha de Confianza Computacional

El Espejismo de Competencia de la IA produce lo que definimos como Brecha de Confianza Computacional [21], [55].

> Definición 3: Brecha de Confianza Computacional. El Brecha de Confianza Computacional Computacional entre la confianza percibida en un resultado generado por IA y el nivel de confiabilidad que puede justificarse por el mecanismo computacional subyacente, la evidencia, el procedimiento de validación o la ruta de ejecución determinista [55]-[21], [55].

El Brecha de Confianza Computacional representa la diferencia entre: [55]-[21], [55]

- la confianza percibida por el usuario humano tras leer una respuesta; y
- el grado de certeza que realmente puede garantizar el mecanismo computacional que lo generó.

Conceptualmente definimos:

$$\text{Computational Trust Gap} = \text{Perceived Confidence} - \text{Computational Guarantee}$$

dónde:

- **La confianza percibida representa el nivel de certeza que un usuario humano atribuye a la respuesta después de evaluar su calidad lingüística.**
- **La Garantía Computacional representa el grado de certeza que puede ser justificado por el proceso computacional subyacente, ya sea determinista, probabilístico o formalmente verificable.**

Esta ecuación no pretende ser una métrica cuantitativa.

Más bien, proporciona un marco conceptual para comprender por qué los usuarios frecuentemente sobreestiman la confiabilidad de los sistemas modernos de IA.

Cuando ambas cantidades son aproximadamente iguales, Brecha de Confianza Computacional se acerca a cero [55]-[21], [55].

Este suele ser el caso de sistemas deterministas como calculadoras, compiladores o algoritmos verificados formalmente.

Por el contrario, cuando la confianza lingüística supera con creces las garantías computacionales, la brecha se vuelve significativa.

Esta situación ocurre con frecuencia al interactuar con modelos de lenguaje grandes sobre cuestiones analíticas, legales, financieras o científicas complejas.

Es importante destacar que los usuarios a menudo no tienen una forma confiable de reconocer cuándo ocurre esta transición.

Por qué esto es importante para los sistemas empresariales

El Brecha de Confianza Computacional tiene consecuencias relativamente limitadas cuando la IA se utiliza para actividades de bajo riesgo como: [55]-[21], [55]

- reunión creativa;
- redacción de documentos;
- traducir texto;
- resumir informes;
- generar prototipos de software;
- escritura creativa.

En estos ámbitos, las imprecisiones ocasionales generalmente acarrean consecuencias operativas limitadas.

Los sistemas empresariales operan bajo requisitos fundamentalmente diferentes.

Las organizaciones rutinariamente toman decisiones que involucran:

- transacciones financieras;
- aprobación de crédito;
- suscripción de seguros;
- cálculo de impuestos;
- diagnóstico médico;
- automatización industrial;
- ciberseguridad;
- informes regulatorios.

En estos contextos, el lenguaje persuasivo no puede sustituir las garantías deterministas.

La confianza que inspira una respuesta es irrelevante a menos que el proceso computacional en sí sea capaz de producir resultados reproducibles, auditables y verificables.

Esta distinción explica por qué las arquitecturas empresariales deberían separar deliberadamente dos responsabilidades fundamentalmente diferentes:

- interpretar la intención humana; y
- ejecutar cálculos deterministas.

Los modelos de lenguaje grandes destacan en la primera tarea.

El software determinista sobresale en el segundo.

Confundir estas responsabilidades inevitablemente aumenta el Brecha de Confianza Computacional y, en consecuencia, el riesgo organizacional [55]-[21], [55].

Más allá del entusiasmo por la IA

Los debates públicos describen con frecuencia el entusiasmo actual por la Inteligencia Artificial como IA Hype.

Si bien esta expresión capta la dimensión social del fenómeno, no explica su mecanismo subyacente.

La cuestión no es simplemente que la sociedad se haya vuelto excesivamente optimista acerca de la IA.

La cuestión es considerablemente más profunda.

Los modelos de lenguaje modernos producen sistemáticamente respuestas que poseen casi todas las características lingüísticas que los humanos han asociado históricamente con la inteligencia, la experiencia y la certeza.

Por lo tanto, el exceso de confianza resultante no es simplemente una consecuencia de la atención de los medios o del marketing.

Surge naturalmente de la interacción entre dos sistemas independientes:

- un modelo computacional probabilístico optimizado para generar un lenguaje plausible; y

un sistema cognitivo humano evolucionó para interpretar el lenguaje fluido como evidencia de conocimiento [21], [19]-[21].

Comprender esta interacción es esencial para la adopción responsable de la Inteligencia Artificial.

El desafío no es reducir el entusiasmo por la IA.

El desafío es aprender a calibrar la confianza según las garantías matemáticas proporcionadas por la arquitectura computacional en lugar de según la calidad persuasiva de su lenguaje.

6. El gran malentendido: la IA no reemplaza al software

Una de las predicciones más comunes en torno a la Inteligencia Artificial es que el software empresarial desaparecerá gradualmente.

Según este punto de vista, las organizaciones ya no necesitarán aplicaciones especializadas porque los usuarios simplemente pedirán a un LLM que realice cualquier tarea que necesiten.

Este artículo sostiene que tal conclusión confunde dos fenómenos fundamentalmente diferentes.

De hecho, la inteligencia artificial está revolucionando la industria del software.

Sin embargo, la interrupción no se produce porque el software se esté volviendo innecesario.

La disrupción se produce porque el software se ha vuelto mucho más fácil de construir.

Esta distinción cambia toda la interpretación económica de la IA moderna.

La pregunta equivocada

Las organizaciones hacen cada vez más preguntas a los proveedores de software, como por ejemplo:

"¿Por qué debería comprar su software si simplemente puedo pedirle a ChatGPT o Claude que realicen la misma tarea?"

Aunque comprensible, esta pregunta se basa en una suposición incorrecta.

La comparación correcta no es:

Software empresarial versus ChatGPT

La comparación correcta es:

Software empresarial versus creación y mantenimiento de una plataforma de software equivalente mediante el desarrollo de software asistido por IA.

Estos son problemas fundamentalmente diferentes.

Un modelo de lenguaje grande puede generar código.

Una plataforma empresarial requiere ingeniería.

El software es mucho más que código

El código fuente representa sólo un componente del software empresarial.

Una plataforma empresarial madura normalmente contiene años (o incluso décadas) de ingeniería acumulada, que incluye:

reglas comerciales; __KEEP[40], [32], [36]-[40]

- algoritmos deterministas;
- arquitectura de base de datos;
- autenticación;
- autorización;
- API;
- motores de flujo de trabajo;
- gestión de transacciones;
- seguridad;
- pistas de auditoría;
- escucha;
- escalabilidad;
- pruebas;
- canales de implementación;
- estrategias de respaldo;
- gestión de versiones;
- cumplimiento normativo;
- documentación;
- experiencia de usuario;
- apoyo operativo.

Estas capacidades no surgen de manera confiable simplemente haciendo una pregunta a Modelo de Lenguaje Grande (LLM).

Deben diseñarse, implementarse, probarse, validarse y mantenerse deliberadamente de forma deliberada.

Un modelo de lenguaje grande puede acelerar significativamente este proceso de ingeniería.

No lo elimina.

La verdadera amenaza competitiva

Por lo tanto, la mayor amenaza competitiva que enfrentan las empresas de software tradicionales no es la IA conversacional en sí.

Es ingeniería de software acelerada por IA [51]-[51].

Los asistentes de codificación modernos como Claude Code, GPT, Codex y sistemas similares han aumentado drásticamente la productividad de los desarrolladores de software.

Las tareas que antes requerían días de implementación frecuentemente pueden completarse en horas.

La refactorización compleja a menudo se puede realizar automáticamente.

La documentación, las pruebas, la depuración y la generación de código se han vuelto sustancialmente más rápidas.

En consecuencia, equipos de ingeniería relativamente pequeños ahora pueden desarrollar plataformas empresariales sofisticadas que antes habrían requerido organizaciones mucho más grandes.

Esto cambia la economía de la creación de software.

No elimina la necesidad del software en sí.

La ventaja competitiva pasa de escribir código a comprender los problemas empresariales.

El nuevo panorama competitivo

Históricamente, una de las mayores ventajas competitivas de las que disfrutaban los principales proveedores de software era la escala de ingeniería.

Se requiere crear aplicaciones empresariales:

- cientos de ingenieros de software;
- grandes equipos de control de calidad;
- equipos de documentación dedicados;
- especialistas en infraestructura;
- años de desarrollo.

La Inteligencia Artificial ha reducido drásticamente esta barrera.

Hoy en día, equipos de ingeniería relativamente pequeños, complementados con IA, pueden alcanzar velocidades de desarrollo que antes eran inalcanzables.

Como consecuencia, la industria del software se está volviendo cada vez más meritocrática.

La ventaja competitiva ya no depende principalmente del número de ingenieros empleados.

En cambio, depende cada vez más de:

- experiencia en el dominio;
- calidad arquitectónica;
- datos de propiedad;

- visión del producto;
- velocidad de ejecución;
- comprensión del cliente.

La capacidad de escribir código se está convirtiendo rápidamente en una mercancía.

La capacidad de resolver problemas empresariales no lo es.

Por qué la mayoría de las organizaciones no deberían crear su propio software empresarial

Una consecuencia importante se desprende naturalmente.

Aunque la IA hace que el desarrollo de software sea mucho más accesible, esto no implica que cada organización deba crear sus propios sistemas empresariales.

Considere una empresa minorista.

Su ventaja competitiva radica en la comercialización, la logística, la experiencia del cliente y la gestión de la cadena de suministro.

No radica en mantener una plataforma CRM.

Similarmente:

Un banco compite a través de la gestión de riesgos y productos financieros.

Una empresa de telecomunicaciones compite a través de la calidad de la red y el servicio al cliente.

Una compañía de seguros compite mediante suscripción y gestión de reclamaciones.

Ninguna de estas organizaciones crea una ventaja competitiva al dedicar recursos de ingeniería a reconstruir software que ya existe.

Incluso si la IA reduce significativamente los costos de desarrollo, la creación y el mantenimiento de software empresarial sigue siendo un compromiso a largo plazo que involucra arquitectura, pruebas, gobernanza, mantenimiento, soporte, ciberseguridad y evolución continua.

El costo de oportunidad a menudo excede el costo de la licencia del software especializado.

En consecuencia, la IA cambia la forma en que se construye el software mucho más que quién debería crearlo.

El nuevo papel del software empresarial

Por lo tanto, el software empresarial se vuelve más valioso, no menos.

Su valor, sin embargo, cambia.

Históricamente, las organizaciones compraban software porque escribirlo era prohibitivamente caro.

En el futuro, las organizaciones comprarán software porque mantenerlo las distrae de su negocio principal.

Esta distinción es fundamental.

El papel del software empresarial evoluciona desde:

Proporcionar funcionalidades que los clientes no pueden crear.

a

Proporcionar funcionalidades que los clientes deciden no crear porque su ventaja competitiva está en otra parte.

La Inteligencia Artificial acelera el desarrollo de software.

No elimina la especialización.

Un nuevo principio económico

Por tanto, la consecuencia económica central de la Inteligencia Artificial se puede resumir de la siguiente manera:

La IA no está reemplazando al software empresarial. La IA está reemplazando gran parte del esfuerzo históricamente requerido para crear software empresarial.

Estas son declaraciones profundamente diferentes.

El primero predice la desaparición del software.

El segundo predice una explosión en la innovación de software impulsada por costos de desarrollo dramáticamente más bajos.

La evidencia sugiere cada vez más que la segunda interpretación describe con mayor precisión la transformación que se está produciendo actualmente.

7. La arquitectura empresarial de próxima generación

Los capítulos anteriores conducen a una conclusión arquitectónica natural.

Los modelos de lenguaje grandes son inherentemente probabilísticos.

Los sistemas empresariales requieren fundamentalmente una ejecución determinista.

Sin embargo, ninguno de estos paradigmas por sí solo es suficiente para resolver problemas empresariales complejos.

Un modelo de lenguaje puede comprender lo que quiere un usuario, pero no posee la experiencia acumulada en el dominio necesaria para resolver correctamente todos los problemas comerciales.

Por el contrario, el software determinista puede ejecutar algoritmos con perfecta reproducibilidad, pero no puede comprender de forma natural la intención humana [30]-[30].

Entre estos dos mundos se encuentra un tercer componente que históricamente ha definido el valor del software empresarial: la inteligencia de dominio.

Por lo tanto, proponemos que la próxima generación de sistemas empresariales no debe entenderse únicamente como la integración de la Inteligencia Artificial con software determinista.

Más bien, deberían verse como arquitecturas compuestas de tres capas computacionales complementarias:

- un Capa de Interfaz Probabilística (PIL) responsable de comprender la intención humana;
- un Capa de Inteligencia de Dominio (DIL) responsable de encapsular el conocimiento empresarial;
- a Capa de Ejecución Determinista (DEL) responsable de ejecutar cálculos confiables.

Cada capa resuelve un problema fundamentalmente diferente.

Juntos definen un nuevo paradigma computacional para sistemas empresariales.

Capa de Interfaz Probabilística (PIL)

La primera capa es responsable de la comunicación.

Históricamente, el software empresarial requería que los usuarios se adaptaran al lenguaje de las máquinas.

Los usuarios aprendieron los menús.

Formularios llenos.

Flujos de trabajo configurados.

Escribió SQL.

Llamadas API.

La Inteligencia Artificial invierte esta relación.

Los humanos ahora se comunican usando su propio idioma.

Por tanto, la responsabilidad de la capa de interfaz probabilística es interpretar las intenciones humanas.

> Definición 5: Capa de Interfaz Probabilística (PIL). La interfaz probabilística LCapa de Interfaz Probabilística (PIL)er que utiliza Modelos de Lenguaje Grandes (LLM) o sistemas probabilísticos de IA relacionados para interpretar la intención del lenguaje natural, resolver ambigüedades, formular planes de candidatos, comunicar explicaciones y enrutar solicitudes sin que ella misma sirva como autoridad final para el cálculo crítico para el negocio.

Las responsabilidades típicas incluyen:

- comprender el lenguaje natural;
- resolver ambigüedades;
- interpretar objetivos;
- responder preguntas;
- generar documentación;
- orquestar flujos de trabajo;
- interactuar conversacionalmente con los usuarios.

Es importante destacar que esta capa es intencionalmente probabilística.

El lenguaje humano en sí es inherentemente ambiguo.

En consecuencia, el objetivo de esta capa no es la corrección determinista.

Su objetivo es comprender lo que el usuario intenta lograr.

Capa de Inteligencia de Dominio (DIL)

Comprender la intención del usuario es sólo el primer paso.

Saber cómo resolver un problema requiere un tipo de inteligencia completamente diferente.

Esta segunda capa representa el verdadero capital intelectual del software empresarial.

Nos referimos a él como Capa de Inteligencia de Dominio (DIL).

> Definición 4: Capa de Inteligencia de Dominio (DIL). La Capa de Inteligencia de Dominio (DIL) codifica conocimiento específico del dominio, reglas comerciales, restricciones regulatorias, métodos analíticos, políticas de decisión, datos de propiedad, procedimientos de validación y experiencia operativa para que la intención interpretada pueda transformarse en un plan computacional gobernado __KEEP[40], [32], [36]-[40].

El DIL contiene el conocimiento acumulado que las organizaciones han desarrollado durante años (o a menudo décadas) de experiencia dentro de un dominio empresarial específico.

Este conocimiento se extiende mucho más allá de la ingeniería de software.

Incluye:

reglas comerciales; __KEEP[40], [32], [36]-[40]

- experiencia en el dominio;
- conocimiento regulatorio;
- metodologías de optimización;
- políticas de decisión;
- mejores prácticas;
- flujos de trabajo propietarios;
- metodologías de ingeniería de características;
- marcos analíticos;
- procedimientos de validación;
- experiencia operativa;

conjuntos de datos propietarios; [30], [29], [30]

- Metodologías de enriquecimiento de datos.

A diferencia de los modelos de lenguaje grande, que poseen un amplio conocimiento general, la capa de inteligencia de dominio contiene conocimiento altamente especializado que diferencia una plataforma empresarial de otra.

Por ejemplo, dos organizaciones pueden utilizar exactamente el mismo modelo de lenguaje.

Sin embargo, producirán soluciones fundamentalmente diferentes porque sus capas de inteligencia de dominio son diferentes.

Esta observación tiene profundas implicaciones.

La ventaja competitiva del software empresarial nunca ha residido principalmente en su código fuente __KEEP[51], [42], [48]-[51].

Más bien, reside en la inteligencia de un dominio específico que se ha codificado progresivamente en software.

En consecuencia, la Inteligencia Artificial reduce drásticamente el coste de escribir código.

No crea automáticamente décadas de experiencia empresarial acumulada.

Capa de Ejecución Determinista (DEL)

Una vez que se ha interpretado la intención del usuario y el conocimiento del dominio ha determinado el curso de acción apropiado, la ejecución se convierte en un problema determinista.

Esta responsabilidad pertenece a la capa de ejecución determinista.

> Definición 6: Capa de Ejecución Determinista (DEL). La ejecución determinista LCapa de Ejecución Determinista (DEL)er que ejecuta procedimientos computacionales validados, flujos de trabajo, transacciones, canales de capacitación de modelos, informes y controles de gobernanza de una manera diseñada para la reproducibilidad, trazabilidad, auditabilidad y responsabilidad operativa [30]-[30].

A diferencia de la interfaz probabilística, cada operación realizada dentro del DEL debe satisfacer los requisitos tradicionales del software empresarial:

reproducibilidad; [30]-[30]

- exactitud;
- auditabilidad;
- trazabilidad;
- seguridad;
- coherencia transaccional;
- cumplimiento normativo.

Las responsabilidades típicas incluyen:

- cálculo matemático;
- algoritmos de optimización;
- ejecución del flujo de trabajo;
- transacciones de bases de datos;
- entrenamiento de modelos;
- despliegue;
- presentación de informes;
- escucha;
- aplicación de la seguridad;
- registro de auditoría.

Dadas entradas idénticas, estado controlado y dependencias estables, el DEL está diseñado para producir resultados reproducibles.

Su propósito no es la flexibilidad.

Su propósito es la confianza.

La separación de responsabilidades

Uno de los principios centrales de esta arquitectura es que cada capa realiza sólo la tarea para la que es naturalmente adecuada.

La capa de interfaz probabilística responde:

¿Qué quiere el usuario?

La capa de inteligencia del dominio responde:

¿Cómo debería resolverse este problema dentro de este ámbito empresarial específico?

La capa de ejecución determinista responde:

Ejecutar la solución de manera reproducible, auditable y matemáticamente confiable.

Esta separación reduce drásticamente el Brecha de Confianza Computacional introducido en el Capítulo 5 [55]-[21], [55].

Los usuarios siguen beneficiándose de la interacción del lenguaje natural mientras que las organizaciones siguen beneficiándose de la ejecución determinista.

Las fortalezas de ambos paradigmas se vuelven complementarias en lugar de competir.

Por qué es importante esta arquitectura

Esta arquitectura de tres capas proporciona varias ventajas importantes.

Primero, las organizaciones se vuelven independientes de cualquier modelo de lenguaje grande en particular.

A medida que evolucionan los modelos básicos, la capa de interfaz probabilística se puede reemplazar sin modificar ni la inteligencia del dominio ni el motor de ejecución determinista.

En segundo lugar, el conocimiento empresarial propietario se separa explícitamente tanto de los modelos lingüísticos como de la infraestructura computacional.

Esto mejora significativamente la mantenibilidad, la gobernanza y la sostenibilidad a largo plazo.

En tercer lugar, la ejecución determinista sigue siendo comprobable, certificable y auditable de forma independiente.

Finalmente, esta arquitectura explica naturalmente por qué el software empresarial sigue existiendo a pesar de la aparición de modelos de lenguaje cada vez más capaces.

La Inteligencia Artificial no reemplaza el software empresarial.

Reemplaza la interfaz tradicional a través de la cual los humanos se comunican con el software empresarial.

La nueva definición de software empresarial

Esta arquitectura sugiere una reinterpretación más amplia del propio software empresarial.

Históricamente, el software ha sido visto a menudo simplemente como colecciones de algoritmos implementados en el código fuente __KEEP[51], [42], [48]-[51].

Esta perspectiva ya no es suficiente.

En cambio, el software empresarial debería entenderse como la codificación de la inteligencia de dominio en procesos computacionales deterministas.

Por tanto, su valor se extiende mucho más allá de la generación de código.

Reside en la acumulación sistemática de conocimiento que permite a las organizaciones resolver problemas altamente especializados de manera confiable, consistente y a escala.

Desde esta perspectiva, los modelos de lenguajes grandes se convierten en interfaces extraordinariamente poderosas.

Domain Intelligence se convierte en la ventaja competitiva de la organización.

La ejecución determinista se convierte en la base de la confianza.

Juntas, estas tres capas definen lo que creemos que constituye la próxima generación de informática empresarial.

Una nueva definición de inteligencia artificial empresarial

Con base en los principios arquitectónicos introducidos a lo largo de este artículo, proponemos la siguiente definición:

La inteligencia artificial empresarial es una arquitectura computacional en la que la inteligencia probabilística interpreta la intención humana, la inteligencia de dominio determina cómo se deben resolver los problemas comerciales y el software determinista ejecuta esas soluciones con confiabilidad matemática, reproducibilidad y gobernanza [30]-[30].

Esta definición difiere fundamentalmente de la percepción cada vez más común de que la IA empresarial consiste simplemente en conectar un modelo de lenguaje grande a los datos organizacionales.

En cambio, reconoce que los sistemas empresariales confiables surgen de la integración de tres formas complementarias de inteligencia:

- **Inteligencia Lingüística, encargada de comprender el lenguaje humano.**
- **Inteligencia de Dominio, encargada de entender el negocio.**
- **Inteligencia Computacional, encargada de ejecutar la computación determinista.**

Sólo la combinación de estas tres capas permite sistemas empresariales que sean simultáneamente intuitivos, confiables, auditables y económicamente sostenibles.

8. Estudio de caso: Aprobación de facturas y adquisición empresarial hasta el pago

Los principios arquitectónicos presentados a lo largo de este documento se vuelven particularmente evidentes en los flujos de trabajo empresariales desde la adquisición hasta el pago, donde la intención del lenguaje natural debe traducirse en una ejecución financiera gobernada.

A primera vista, los modelos modernos de lenguaje grande parecen capaces de realizar muchas actividades asociadas con los equipos de adquisiciones y cuentas por pagar.

Resumen las facturas.

Redactan solicitudes de compra.

Explican el lenguaje del contrato.

Responden preguntas de proveedores.

Generan notas de aprobación.

Clasifican los gastos.

Naturalmente, esto lleva a muchas organizaciones a plantearse una pregunta tentadora:

Si un LLM puede comprender una solicitud de adquisición y producir una recomendación fluida, ¿por qué una empresa seguiría necesitando una plataforma de adquisición a pago?

La respuesta está en la diferencia entre interpretar una solicitud comercial y ejecutar una transacción financiera gobernada.

Estos son desafíos fundamentalmente diferentes.

Interpretar una factura no es aprobar un pago

Aprobar una factura u orden de compra no es simplemente una tarea lingüística. Es un proceso empresarial controlado que debe satisfacer requisitos financieros, contractuales, operativos y de auditoría.

Un sistema de adquisición a pago de nivel de producción debe proporcionar capacidades deterministas tales como:

validación de proveedores;

conciliación de órdenes de compra;

extracción y normalización de datos de facturas;

comprobaciones de disponibilidad presupuestaria;

cumplimiento del límite de aprobación;

controles de segregación de funciones;

validación fiscal y jurisdiccional;

cumplimiento de contratos;

detección de facturas duplicadas;

programación de pagos;

registro de auditoría;

Conciliación de ERP;

control de seguridad y acceso;

cumplimiento normativo.

Un modelo de lenguaje puede ayudar en cada paso individual. No puede garantizar que el flujo de trabajo completo siga siendo consistente, autorizado, reproducible y auditable a lo largo de los años de operación empresarial.

Esa responsabilidad pertenece al software determinista.

Aplicar la arquitectura de tres capas

La adquisición hasta el pago proporciona un claro ejemplo de cómo surge naturalmente la arquitectura de tres capas propuesta en este artículo.

Capa de Interfaz Probabilística (PIL)

El empleado, comprador, analista financiero o ejecutivo se comunica con el sistema mediante lenguaje natural.

Las solicitudes típicas incluyen:

"Aprobar esta factura si coincide con la orden de compra."

"Crear una solicitud de compra para este proveedor."

"Explique por qué este pago está bloqueado".

"Renovar este contrato de proveedor si los términos aún se cumplen".

El modelo de lenguaje grande interpreta estas intenciones.

Aún no se ha producido ninguna ejecución empresarial determinista.

Capa de Inteligencia de Dominio (DIL)

Una vez que se ha comprendido el objetivo del usuario, el sistema aplica la experiencia acumulada en adquisiciones, finanzas y cumplimiento.

Los ejemplos incluyen:

políticas de riesgo de proveedores;

reglas de control presupuestario;

matrices de aprobación;

obligaciones contractuales;

tratamiento fiscal;

condiciones de pago;

controles internos;

requisitos de auditoría;

configuración del ERP;

procedimientos operativos específicos de la organización.

Este conocimiento generalmente se ha acumulado a través de años de experiencia operativa.

Por lo general, no se puede producir de manera confiable solicitando un Modelo de Lenguaje Grande (LLM) de forma aislada.

Representa la inteligencia empresarial integrada en la plataforma.

Capa de Ejecución Determinista (DEL)

Finalmente, el software determinista ejecuta el flujo de trabajo financiero solicitado.

Las responsabilidades típicas incluyen:

cotejar facturas con órdenes de compra;

validar datos maestros de proveedores;

comprobar la disponibilidad presupuestaria;

aprobaciones de rutas;

hacer cumplir los derechos de acceso;

bloquear violaciones de políticas;

publicar transacciones en el ERP;

programación de pagos;

registrar pistas de auditoría inmutables;

conciliar registros financieros.

Dadas las mismas facturas, políticas, estados de aprobación y configuraciones del sistema, esta capa debe producir resultados consistentes.

Su comportamiento es determinista por diseño.

Por qué es importante la auditabilidad

Las adquisiciones empresariales y las cuentas por pagar operan en entornos donde las decisiones financieras deben poder explicarse a posteriori.

Las organizaciones necesitan reconstruir rutinariamente por qué se aprobó una factura, quién la aprobó, qué política se aplicó, si el proveedor era válido y si la transacción respetó las restricciones presupuestarias, contractuales y de cumplimiento.

Esto implica que cada acción financiera debe ser rastreable, incluyendo:

solicitar origen;

identidad del proveedor;

vinculación de órdenes de compra;

cadena de aprobación;

evaluación de políticas;

estado del presupuesto;

referencia del contrato;

ejecución del pago.

Los modelos de lenguaje grandes no están diseñados para proporcionar este nivel de auditabilidad empresarial determinista.

Su objetivo es ayudar a la interpretación y comunicación humana.

Existen plataformas empresariales deterministas para garantizar una ejecución controlada.

Estos objetivos son complementarios en lugar de competir.

La importancia de la inteligencia de dominio

Quizás el error más importante en torno a las adquisiciones asistidas por IA es la creencia de que la interpretación de documentos constituye la principal fuente de valor.

En realidad, la interpretación es sólo la interfaz.

La verdadera ventaja competitiva reside en la inteligencia de dominio acumulada integrada en la plataforma de adquisiciones y finanzas.

Los ejemplos incluyen:

metodologías de calificación de proveedores;

diseño de políticas de aprobación;

procedimientos de cumplimiento de contratos;

reglas de puntuación de riesgos;

métodos de gobernanza presupuestaria;

flujos de trabajo específicos de la organización;

datos de proveedores propietarios;

prácticas de control financiero.

Un modelo de lenguaje puede generar la correspondiente explicación o sugerencia de flujo de trabajo.

No posee automáticamente la disciplina operativa necesaria para decidir qué regla debe aplicarse bajo qué condiciones comerciales.

Esa experiencia pertenece a la capa de inteligencia del dominio.

Más allá de la adquisición hasta el pago

Aunque este estudio de caso se centra en las adquisiciones y la aprobación de facturas, el mismo principio arquitectónico se generaliza naturalmente a prácticamente todos los dominios empresariales.

Gestión de relaciones con los clientes.

Planificación de recursos empresariales.

Optimización de la cadena de suministro.

Ciberseguridad.

Diagnóstico médico.

Cumplimiento legal.

Automatización industrial.

En todos los casos:

La Inteligencia Artificial entiende lo que quiere el usuario.

Domain Intelligence determina cómo la organización resuelve esa clase de problemas.

El software determinista ejecuta los procesos de negocio correspondientes.

La arquitectura sigue siendo idéntica.

Sólo cambia el conocimiento del dominio.

La propuesta de valor empresarial

Este análisis conduce a una conclusión importante.

La propuesta de valor del software empresarial ya no es principalmente la automatización de la computación.

La computación en sí está cada vez más mercantilizada.

En cambio, el software empresarial crea valor al integrar tres capacidades que son extremadamente difíciles de replicar simultáneamente:

interacción humana intuitiva a través de modelos de lenguaje probabilísticos;

experiencia acumulada en el campo codificada en políticas, flujos de trabajo y metodologías reutilizables;

ejecución determinista capaz de satisfacer los requisitos empresariales.

Por lo tanto, las organizaciones compran software empresarial no porque carezcan de acceso a la Inteligencia Artificial.

Compran software empresarial porque crear, validar y mantener continuamente esta arquitectura integrada desviaría recursos de las actividades que realmente diferencian su negocio.

9. Trabajo relacionado

La teoría propuesta se sitúa dentro de varias tradiciones de investigación maduras y no fuera de ellas. El trabajo fundamental en teoría de la información, inteligencia artificial simbólica, razonamiento probabilístico y aprendizaje automático moderno establece la distinción entre comunicación, computación, búsqueda, incertidumbre y representación aprendida __KE[9]0__–[9]. Las arquitecturas de transformadores y los modelos básicos proporcionan la base técnica para Modelos de Lenguaje Grandes (LLM) contemporáneo, incluido el entrenamiento previo, el ajuste de instrucciones y la alineación a través de retroalimentación humana o[16] [10]-[16].

Trabajos anteriores también han examinado las limitaciones de Modelos de Lenguaje Grandes (LLM), incluidas las alucinaciones, la generación estocástica del lenguaje, la calibración y los riesgos sociales de confundir texto fluido con conocimiento confiable [55], [18], [55]. Estos resultados apoyan la distinción del artículo entre plausibilidad y verdad, pero no proporcionan por sí mismos una arquitectura empresarial para asignar modelos de lenguaje probabilísticos, conocimiento de dominio y ejecución determinista para separar responsabilidades del sistema.

El Espejismo de Competencia de la IA y el Brecha de Confianza Computacional están relacionados con el trabajo sobre la confianza en la automatización, el uso indebido y la dependencia excesiva, la confianza adecuada y la calibración humana de los sistemas automatizados [21]-[21]. La contribución de este artículo es reinterpretar esas preocupaciones en el contexto específico de Modelos de Lenguaje Grandes (LLM): la competencia percibida surge de un lenguaje probabilístico fluido, mientras que la brecha surge porque la confianza lingüística y las garantías computacionales se generan mediante diferentes mecanismos.

Las propuestas Capa de Inteligencia de Dominio (DIL) y Capa de Ejecución Determinista (DEL) se superponen con la investigación en arquitectura de software, modularidad, arquitectura de aplicaciones empresariales y marcos de arquitectura empresarial [38]-[38]. También están relacionados con la investigación sobre aprendizaje automático de producción, MLOps, gobernanza de modelos, validación de datos, tarjetas modelo y do[30]ntation [22]-[30]. La distinción introducida aquí es arquitectónica más que meramente operativa: el artículo sostiene que el conocimiento del dominio debe tratarse como una capa computacional de primera clase entre la interpretación y la ejecución de la intención.

El artículo también coincide con la literatura sobre rediseño de procesos de negocios, ventaja competitiva, programación como construcción de teorías y diseño de sistemas humano-computadora [42]-[42]. Estos trabajos ayudan a explicar por qué el valor del software a menudo reside en el conocimiento organizacional codificado y no solo en el código. En este sentido, Capa de Inteligencia de Dominio (DIL) está relacionado con descripciones anteriores de conocimiento de procesos de negocio y conocimiento de diseño de software, pero está enmarcado específicamente para sistemas empresariales que utilizan Modelos de Lenguaje Grandes (LLM) como interfaces probabilísticas.

Finalmente, la investigación sobre agentes inteligentes, generación de recuperación aumentada, uso de herramientas, bucles de razonamiento y acción y programación asistida por IA demuestra que Modelos de Lenguaje Grandes (LLM) puede participar en flujos de trabajo cada vez más complejos [54]-[54]. Este artículo no niega esos desarrollos. Su afirmación es más limitada: incluso cuando los agentes o los sistemas de generación de código son poderosos, la confianza empresarial aún requiere una separación explícita entre interpretación probabilística, inteligencia de dominio y ejecución determinista.

Por lo tanto, la novedad de Computación Orientada por Intención (IDC) no radica en la observación aislada de que los usuarios pueden expresar su intención, ni en la existencia de agentes, flujos de trabajo, capas semánticas o arquitectura empresarial. El trabajo relacionado en redes basadas en intenciones y gestión de recursos impulsadas por intenciones también estudia la traducción de los objetivos de alto nivel a la configuración operativa. La contribución que se reclama aquí es la teoría integrada de la computación empresarial que combina la arquitectura Espejismo de Competencia de la IA, Brecha de Confianza Computacional y PIL -> DIL -> DEL como una explicación unificada de cómo Modelos de Lenguaje Grandes (LLM) debe posicionarse dentro de sistemas empresariales confiables.

10. Limitaciones

Este artículo es conceptual. Propone una teoría y un marco arquitectónico, pero no valida experimentalmente Capa de Interfaz Probabilística (PIL), Capa de Inteligencia de Dominio (DIL) y Capa

de Ejecución Determinista (DEL) en un entorno empírico controlado. El trabajo futuro debería evaluar si los sistemas diseñados de acuerdo con esta arquitectura mejoran de manera mensurable la confiabilidad, la auditabilidad, la calibración del usuario, la productividad del desarrollo o los resultados operativos.

El artículo no compara la arquitectura propuesta con la IA neurosimbólica, los métodos formales, la programación probabilística, el análisis semántico, la generación de recuperación aumentada o los sistemas de flujo de trabajo agente en un punto de referencia sistemático. Estos enfoques pueden abordar parcialmente algunas de las mismas preocupaciones a través de diferentes mecanismos y deberían examinarse en trabajos futuros.

El artículo no afirma que Modelos de Lenguaje Grandes (LLM) sea incapaz de razonar. La investigación contemporánea ha demostrado que los métodos orientados a la estimulación, el uso de herramientas, la recuperación y el razonamiento pueden mejorar sustancialmente el desempeño en tareas complejas [54]-[47], [52]-[54]. En cambio, la afirmación es que los sistemas empresariales no deberían tratar la generación probabilística del lenguaje únicamente como un sustituto del conocimiento del dominio gobernado y la ejecución determinista.

El documento tampoco invalida a los agentes autónomos de IA. Los agentes pueden convertirse en componentes importantes de los sistemas empresariales. La limitación es que la autonomía del agente no elimina la necesidad de límites arquitectónicos explícitos, observabilidad, gobernanza, reproducibilidad y responsabilidad.

Finalmente, los conceptos propuestos requieren una mayor operacionalización. El trabajo futuro debe definir indicadores mensurables para Brecha de Confianza Computacional, desarrollar patrones de diseño para implementar Capa de Inteligencia de Dominio (DIL) y comparar sistemas PIL -> DIL -> DEL con arquitecturas alternativas en dominios como finanzas, atención médica, optimización de la cadena de suministro, ciberseguridad y aprendizaje automático regulado.

11. Conclusión

La aparición de los Grandes Modelos de Lenguaje representa uno de los avances tecnológicos más significativos en la historia de la informática.

Nunca antes las computadoras habían sido capaces de comprender y generar lenguaje natural con tanta fluidez, flexibilidad y conciencia contextual.

Su impacto en la ingeniería de software, el trabajo del conocimiento y la interacción persona-computadora sin duda remodelará prácticamente todas las industrias en las próximas décadas.

Sin embargo, esta transformación no debe interpretarse como la sustitución del software empresarial por la Inteligencia Artificial conversacional.

A lo largo de este artículo hemos argumentado que esta interpretación generalizada surge de un malentendido fundamental de la naturaleza computacional de los modelos de lenguaje grande.

Los LLM son sistemas probabilísticos.

El software empresarial es determinista.

Estos no son paradigmas en competencia.

Resuelven diferentes clases de problemas.

Los modelos de lenguaje sobresalen en la interpretación de las intenciones humanas, reduciendo la ambigüedad y facilitando la comunicación entre humanos y máquinas.

El software determinista sobresale en la ejecución de reglas de negocio, cálculo matemático, optimización, gobernanza, auditoría y toma de decisiones reproducible __KEEP[40], [32], [36]-[40].

Ningún paradigma puede reemplazar completamente al otro.

Más bien, el futuro de la informática empresarial reside en su integración deliberada.

Para formalizar esta perspectiva, este artículo introdujo una arquitectura computacional de tres capas que consta de:

- el Capa de Interfaz Probabilística (PIL), responsable de comprender el lenguaje humano;
- el Capa de Inteligencia de Dominio (DIL), responsable de encapsular la experiencia empresarial y el conocimiento organizacional acumulado;
- el Capa de Ejecución Determinista (DEL), responsable de ejecutar procesos computacionales confiables.

Juntas, estas capas separan tres formas fundamentalmente diferentes de inteligencia que a menudo han sido tratadas incorrectamente como un solo problema:

entender el lenguaje,

entender el negocio,

y ejecutar el cálculo.

Reconocer estas distinciones cambia fundamentalmente cómo se debe entender el software empresarial.

Históricamente, el software ha sido visto principalmente como código fuente __KEEP[51], [42], [48]-[51].

La Inteligencia Artificial ahora demuestra que el código fuente se está convirtiendo cada vez más en un producto básico __KEEP[51], [42], [48]-[51].

Lo que sigue siendo difícil de replicar no es la capacidad de escribir software.

Es la capacidad de codificar décadas de experiencia en el campo en sistemas deterministas capaces de producir resultados comerciales confiables, reproducibles y auditables.

En consecuencia, la verdadera ventaja competitiva del software empresarial está cambiando.

Cada vez más, el valor residirá menos en la ingeniería de software en sí y más en la codificación sistemática de la inteligencia de dominio.

La inteligencia artificial reduce drásticamente el costo de creación de software.

No crea automáticamente el conocimiento empresarial necesario para resolver problemas altamente especializados.

Esta distinción también explica por qué muchas discusiones actuales sobre el futuro del software llegan a conclusiones engañosas.

Las organizaciones no abandonarán las plataformas empresariales porque exista la IA conversacional.

En cambio, las plataformas de software incorporarán cada vez más la IA conversacional y al mismo tiempo seguirán brindando la ejecución determinista, la gobernanza y la experiencia en el dominio que las organizaciones requieren.

La interfaz cambia.

Las bases computacionales permanecen.

Por lo tanto, la industria del software enfrenta una transformación profunda, pero no la que comúnmente se describe.

La principal disrupción no es que la Inteligencia Artificial reemplace al software.

La principal disrupción es que la Inteligencia Artificial reduce radicalmente el coste de creación de software.

A medida que las barreras al desarrollo de software sigan cayendo, la competencia se desplazará cada vez más desde la escala de ingeniería hacia la experiencia en el dominio, la calidad arquitectónica y la capacidad de convertir el conocimiento organizacional en sistemas computacionales reutilizables.

En este sentido, la Inteligencia Artificial se asemeja a revoluciones tecnológicas anteriores que redujeron drásticamente los costos de producción sin eliminar los productos en sí.

Así como la automatización industrial transformó la fabricación sin eliminar los productos manufacturados, la Inteligencia Artificial está transformando la ingeniería de software sin eliminar el software.

Finalmente, este artículo propone una reinterpretación más amplia del papel de la Inteligencia Artificial dentro de los sistemas empresariales.

En lugar de ver la IA como un reemplazo del software determinista, sostenemos que debe entenderse como la interfaz de lenguaje natural a través de la cual los humanos se comunican con sistemas computacionales deterministas enriquecidos por inteligencia de dominio específico.

Desde esta perspectiva, la Inteligencia Artificial no representa el fin del software empresarial.

Representa el comienzo de una nueva generación de informática empresarial.

Uno en el que el lenguaje se convierte en la interfaz universal.

La inteligencia de dominio se convierte en el principal activo estratégico.

Y la computación determinista sigue siendo la base de la confianza.

---

12. Principios propuestos

Los argumentos presentados a lo largo de este artículo pueden resumirse en los siguientes principios:

Los modelos de lenguaje grandes optimizan la plausibilidad, no la verdad [18], [16]-[18].

1. El lenguaje fluido no debe confundirse con la certeza computacional.

El Brecha de Confianza Computacional aumenta siempre que la confianza percibida excede las garantías computacionales [55]-[21], [55].

2. La Inteligencia Artificial está transformando el desarrollo de software mucho más de lo que lo está reemplazando.

El software empresarial obtiene su valor principalmente de la inteligencia de dominio codificada y no solo del código fuente __KEEP[51], [42], [48]-[51].

3. Una IA empresarial confiable requiere la separación explícita de la comprensión del lenguaje probabilístico, la inteligencia de dominio y la computación determinista.
4. El futuro de la informática empresarial pertenece a las arquitecturas que integran estas tres formas complementarias de inteligencia.

13. Referencias

[1] C. E. Shannon, “Una teoría matemática de la comunicación”, Bell System Technical Journal, vol. 27, núm. 3, págs. 379–423, 1948; vol. 27, núm. 4, págs. 623–656, 1948.

[2] A. M. Turing, “Maquinaria informática e inteligencia”, Mind, vol. 59, núm. 236, págs. 433–460, 1950.

[3] J. McCarthy, M. L. Minsky, N. Rochester y C. E. Shannon, “Una propuesta para el proyecto de investigación de verano de Dartmouth sobre inteligencia artificial”, 1955.

[4] A. Newell y H. A. Simon, “La informática como investigación empírica: símbolos y búsqueda”, Comunicaciones del ACM, vol. 19, núm. 3, págs. 113-126, 1976.

[5] H. A. Simon, Las ciencias de lo artificial, 3ª ed. Cambridge, MA, EE.UU.: MIT Press, 1996.

[6] J. Pearl, Razonamiento probabilístico en sistemas inteligentes. San Mateo, California, Estados Unidos: Morgan Kaufmann, 1988.

[7] J. Pearl, Causalidad: modelos, razonamiento e inferencia, 2ª ed. Cambridge, Reino Unido: Cambridge University Press, 2009.

[8] S. Russell y P. Norvig, Inteligencia artificial: un enfoque moderno, 4ª ed. Hoboken, Nueva Jersey, EE. UU.: Pearson, 2020.

[9] Y. LeCun, Y. Bengio y G. Hinton, “Aprendizaje profundo”, Nature, vol. 521, págs. 436–444, 2015.

[10] A. Vaswani et al., “La atención es todo lo que necesita”, en Advances in Neural Information Processing Systems, 2017, págs. 5998–6008.

[11] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, “BERT: Entrenamiento previo de transformadores bidireccionales profundos para la comprensión del lenguaje”, en Proc. NAACL-HLT, 2019, págs. 4171–4186.

[12] T. B. Brown et al., “Los modelos lingüísticos son aprendices de pocas oportunidades”, en Advances in Neural Information Processing Systems, vol. 33, 2020, págs. 1877-1901.

[13] OpenIA, “Informe técnico GPT-4”, arXiv:2303.08774, 2023.

[14] L. Ouyang et al., “Entrenamiento de modelos de lenguaje para seguir instrucciones con retroalimentación humana”, en Advances in Neural Information Processing Systems, vol. 35, 2022, págs. 27730–27744.

[15] Y. Bai et al., “IA constitucional: inofensividad a partir de la retroalimentación de la IA”, arXiv:2212.08073, 2022.

[16] R. Bommasani et al., “Sobre las oportunidades y riesgos de los modelos de fundaciones”, arXiv:2108.07258, 2021.

[17] E. M. Bender, T. Gebru, A. McMillan-Major y S. Shmitchell, "Sobre los peligros de los loros estocásticos: ¿Pueden los modelos de lenguaje ser demasiado grandes?" en Proc. ACM HECHO, 2021, págs. 610–623.

[18] Z. Ji et al., “Encuesta sobre alucinaciones en la generación del lenguaje natural”, ACM Computing Surveys, vol. 55, núm. 12, págs. 1 a 38, 2023.

[19] R. Parasuraman y V. Riley, “Los seres humanos y la automatización: uso, mal uso, desuso, abuso”, Factores humanos, vol. 39, núm. 2, págs. 230-253, 1997.

[20] J. D. Lee y K. A. Consulte “Confianza en la automatización: diseño para una confianza adecuada”, Human Factors, vol. 46, núm. 1, págs. 50 a 80, 2004.

[21] K. A. Hoff y M. Bashir, “Confianza en la automatización: integración de evidencia empírica sobre factores que influyen en la confianza”, Factores humanos, vol. 57, núm. 3, págs. 407–434, 2015.

[22] D. Sculley et al., “Deuda técnica oculta en sistemas de aprendizaje automático”, en Advances in Neural Information Processing Systems, 2015, págs. 2503–2511.

[23] E. Breck, S. Cai, E. Nielsen, M. Salib y D. Sculley, “La puntuación de la prueba de ML: una rúbrica para la preparación de la producción de ML y la reducción de la deuda técnica”, en Proc. IEEE BigData, 2017, págs. 1123–1132.

[24] S. Amershi et al., “Ingeniería de software para el aprendizaje automático: un estudio de caso”, en Proc. CISE-SEIP, 2019, págs. 291–300.

[25] E. Breck et al., “Validación de datos para el aprendizaje automático”, en Proc. SysML, 2019.

[26] D. Baylor et al., “TFX: una plataforma de aprendizaje automático a escala de producción basada en TensorFlow”, en Proc. KDD, 2017, págs. 1387-1395.

[27] M. Zaharia et al., “Acelerar el ciclo de vida del aprendizaje automático con MLflow”, IEEE Data Engineering Bulletin, vol. 41, núm. 4, págs. 39–45, 2018.

[28] D. Kreuzberger, N. Kühl y S. Hirschl, “Operaciones de aprendizaje automático (MLOps): descripción general, definición y arquitectura”, IEEE Access, vol. 11, págs. 31866–31879, 2023.

[29] M. Mitchell et al., “Tarjetas modelo para informes modelo”, en Proc. ACM HECHO, 2019, págs. 220–229.

[30] T. Gebru et al., “Hojas de datos para conjuntos de datos”, Comunicaciones del ACM, vol. 64, núm. 12, págs. 86–92, 2021.

[31] D. Garlan y M. Shaw, “Una introducción a la arquitectura de software”, en Advances in Software Engineering and Knowledge Engineering, vol. 1, 1993, págs. 1–39.

[32] D. L. Parnas, “Sobre los criterios que se utilizarán al descomponer sistemas en módulos”, Comunicaciones de ACM, vol. 15, núm. 12, págs. 1053-1058, 1972.

[33] F. P. Brooks, Jr., “No hay solución milagrosa: esencia y accidentes de la ingeniería de software”, Computadora, vol. 20, núm. 4, págs. 10 a 19, 1987.

[34] B. W. Boehm, “Un modelo en espiral de desarrollo y mejora de software”, Computer, vol. 21, núm. 5, págs. 61–72, 1988.

[35] L. Bass, P. Clements y R. Kazman, Arquitectura de software en la práctica, 4ª ed. Boston, MA, EE. UU.: Addison-Wesley, 2021.

[36] M. Fowler, Patrones de arquitectura de aplicaciones empresariales. Boston, MA, EE.UU.: Addison-Wesley, 2002.

[37] J. A. Zachman, “Un marco para la arquitectura de sistemas de información”, IBM Systems Journal, vol. 26, núm. 3, págs. 276–292, 1987.

[38] El Grupo Abierto, Estándar TOGAF, 10ª ed. San Francisco, CA, EE. UU.: Grupo Abierto, 2022.

[39] T. H. Davenport y J. E. Short, “La nueva ingeniería industrial: tecnología de la información y rediseño de procesos comerciales”, Sloan Management Review, vol. 31, núm. 4, págs. 11 a 27, 1990.

[40] M. E. Porter y V. E. Millar, “Cómo la información proporciona una ventaja competitiva”, Harvard Business Review, vol. 63, núm. 4, págs. 149-160, 1985.

[41] M. Winograd y F. Flores, Comprensión de las computadoras y la cognición: una nueva base para el diseño. Norwood, Nueva Jersey, EE.UU.: Ablex, 1986.

[42] F. Naur, “La programación como construcción de teorías”, Microprocesamiento y microprogramación, vol. 15, núm. 5, págs. 253–261, 1985.

[43] F. M. T. Brazier et al., “Modelado organizacional basado en agentes para la integración empresarial”, en Proc. HICSS, 1998.

[44] M. Wooldridge y N. R. Jennings, “Agentes inteligentes: teoría y práctica”, The Knowledge Engineering Review, vol. 10, núm. 2, págs. 115-152, 1995.

[45] P. Lewis et al., “Generación de recuperación aumentada para tareas de PNL con uso intensivo de conocimiento”, en Advances in Neural Information Processing Systems, vol. 33, 2020, págs. 9459–9474.

[46] S. Yao et al., “ReAct: Sinergizar el razonamiento y la actuación en modelos de lenguaje”, en Proc. ICLR, 2023.

[47] T. Schick et al., “Toolformer: Los modelos de lenguaje pueden aprender a usar herramientas por sí solos”, en Advances in Neural Information Processing Systems, vol. 36, 2023.

[48] M. Chen et al., “Evaluación de modelos de lenguaje grandes entrenados en código”, arXiv:2107.03374, 2021.

[49] Y. Li et al., “Generación de código a nivel de competencia con AlphaCode”, Science, vol. 378, núm. 6624, págs. 1092-1097, 2022.

[50] P. Vaithilingam, T. Zhang y E. L. Glassman, “Expectativa versus experiencia: evaluación de la usabilidad de herramientas de generación de código impulsadas por grandes modelos de lenguaje”, en Extended Abstracts of CHI, 2022.

[51] S. Peng, E. Kalliamvakou, P. Cihon y M. Demirer, “El impacto de la IA en la productividad de los desarrolladores: evidencia de GitHub Copilot”, arXiv:2302.06590, 2023.

[52] J. Wei et al., “La cadena de pensamiento provoca razonamiento en modelos de lenguaje grandes”, en Advances in Neural Information Processing Systems, vol. 35, 2022, págs. 24824–24837.

[53] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo e Y. Iwasawa, “Los modelos de lenguaje grandes son razonadores de tiro cero”, en Advances in Neural Information Processing Systems, vol. 35, 2022, págs. 22199–22213.

[54] X. Wang et al., “La autoconsistencia mejora el razonamiento en cadena de pensamiento en modelos de lenguaje”, en Proc. ICLR, 2023.

[55] S. Kadavath et al., “Los modelos lingüísticos (en su mayoría) saben lo que saben”, arXiv:2207.05221, 2022.

[56] TM Mitchell, Aprendizaje automático. Nueva York, NY, Estados Unidos: McGraw-Hill, 1997.