¿Cómo aprende a hablar una máquina?

Irina Arévalo[1]

Departamento de matemática e informática aplicadas a las ingenierías civil y naval
Universidad Politécnica de Madrid

1. Introducción

Durante mucho tiempo, hablar fue una frontera. Las máquinas calculaban, almacenaban datos, seguían instrucciones, pero el lenguaje parecía otra cosa: demasiado ambiguo, demasiado flexible, demasiado humano. Entender un chiste, completar una frase, responder a una pregunta abierta o resumir un texto parecían tareas reservadas a la inteligencia natural.

De hecho, esa frontera está en el corazón mismo de la historia de la inteligencia artificial [6]. En 1950, Alan Turing se planteó una pregunta que acabaría siendo fundacional: ¿pueden pensar las máquinas? Para esquivar la dificultad de definir qué significa exactamente "pensar", propuso cambiar la pregunta por otra más concreta: si una máquina conversa de manera indistinguible de una persona, ¿no deberíamos atribuirle algún tipo de inteligencia? Durante décadas, esa idea pareció casi una provocación filosófica. Hoy, en cambio, convivimos con sistemas que escriben, responden y dialogan con una soltura que Turing habría encontrado, como poco, fascinante.

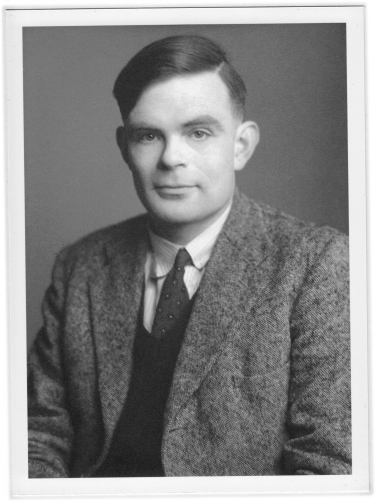

Figura 1: Alan Turing, al que ya conocimos en el *Retazos* sobre Criptografía

A estos sistemas se les suele llamar LLMs, siglas de *Large Language Model*, es decir, modelo de lenguaje grande. El nombre no está mal elegido: son "modelos de lenguaje",

[1] irina.arevalo@upm.es

porque trabajan con palabras, frases y textos. Y son "grandes" porque contienen una cantidad gigantesca de parámetros: números que se ajustan durante el entrenamiento hasta que el sistema aprende a producir texto coherente. En cierto sentido, un LLM es una enorme máquina matemática especializada en predecir qué palabra (o, más exactamente, qué fragmento de palabra) tiene sentido que venga después de otra.

Dicho así, la idea parece decepcionante. ¿De verdad todo este despliegue tecnológico consiste solo en "adivinar la siguiente palabra"? Sorprendentemente, sí. Pero precisamente ahí está una de las lecciones más hermosas de las matemáticas: a veces una regla simple, repetida a gran escala, puede producir comportamientos extraordinariamente complejos.

No sería la primera vez. A partir de unas pocas leyes, Newton explicó el movimiento de los planetas. A partir de reglas locales muy sencillas pueden aparecer estructuras riquísimas en la teoría de sistemas dinámicos. Y a partir de ceros y unos un ordenador ejecuta videojuegos, simulaciones o videollamadas. En matemáticas ocurre a menudo que bajo fenómenos muy sofisticados se esconde una idea elemental.

En el caso de los LLMs, la pregunta central es la siguiente: ¿cómo se transforma el lenguaje en un objeto matemático? Porque una máquina no entiende las palabras como nosotros. No sabe qué es un perro, ni una integral, ni la lluvia en una tarde de noviembre. Lo único que puede manipular son números. Si queremos que trabaje con lenguaje, primero hay que hacer algo que a primera vista parece imposible: convertir palabras en números sin perder del todo su significado.

Ese es el primer truco. Y después viene el segundo: conseguir que, a partir de esos números, la máquina aprenda patrones, relaciones, dependencias gramaticales y hasta ciertas regularidades del mundo. Todo ello sin que nadie le escriba a mano una gramática completa del español, una lista infinita de reglas semánticas o una enciclopedia universal.

Lo fascinante de este proceso es que en él entran en juego muchas ideas matemáticas a la vez: vectores, distancias, probabilidad, optimización, geometría en muchas dimensiones, funciones que reparten pesos, productos matriciales... Los LLMs son, en el fondo, una gran historia de cómo el álgebra lineal y la probabilidad se las arreglan para dar forma a algo tan aparentemente etéreo como el lenguaje.

2. El problema de convertir palabras en números

Supongamos que queremos enseñar a una máquina a trabajar con la frase:

> "Las matemáticas ayudan a entender el mundo."

Para nosotros es una oración natural. Para un ordenador, en cambio, no es más que una secuencia de símbolos. No sabe qué hacer con "matemáticas" ni con "mundo" mientras sigan siendo solo letras.

La primera idea podría ser asignar un número a cada palabra. Por ejemplo:

- "las" $\to$ 1
- "matemáticas" $\to$ 2
- "ayudan" $\to$ 3
- "a" $\to$ 4
- "entender" $\to$ 5
- "el" $\to$ 6

- “mundo” $\to$ 7

Así, la frase quedaría convertida en una secuencia de enteros. Parece un comienzo razonable, pero enseguida surgen problemas. El primero es que el número 2 no se parece más a “física” que a “bocadillo”, aunque ambas palabras puedan pertenecer a contextos parecidos. La numeración simple sirve para identificar palabras, pero no para reflejar relaciones entre ellas.

Además, el lenguaje es enorme. Aparecen nombres propios, errores ortográficos, siglas, neologismos, tecnicismos. ¿Hay que tener una lista cerrada con todas las palabras posibles? Eso sería inviable.

Por eso los LLMs no trabajan exactamente con palabras completas, sino con *tokens*, pequeñas piezas de texto que pueden ser palabras enteras, sílabas, prefijos, sufijos o fragmentos frecuentes. Por ejemplo, la palabra “matemáticamente” podría descomponerse en varias partes, como “matemática” y “-mente”, o incluso en fragmentos aún más pequeños, dependiendo del sistema de tokenización empleado. Esto permite manejar mejor vocabularios enormes y reutilizar piezas conocidas para palabras nuevas.

Pero seguimos teniendo el mismo problema de fondo: identificar no es comprender. Para una máquina, no basta saber que el token “triángulo” es el número 18427. Necesita una representación más rica, algo que refleje cómo se usa esa palabra en relación con otras.

Y aquí entra una idea de una elegancia casi geométrica: representar cada token mediante un vector, es decir, una lista larga de números reales. Esta manera de representar palabras como vectores aprendidos a partir de grandes colecciones de texto dio lugar a avances decisivos en procesamiento del lenguaje natural [4].

En lugar de decir que “triángulo” es simplemente el número 18427, el modelo lo representa, por ejemplo, como algo del estilo:

$$(0{,}12,\ -0{,}44,\ 1{,}03,\ \dots)$$

No importa aquí la longitud exacta del vector. Lo importante es la idea: cada palabra o fragmento de palabra pasa a ocupar una posición en un espacio de muchas dimensiones. En ese espacio, algunas palabras quedan más cerca y otras más lejos. Ya no se trata solo de etiquetar, sino de situar geométricamente.

Esta decisión es crucial. Porque, una vez que las palabras viven en un espacio geométrico, podemos empezar a hacer matemáticas con ellas. Podemos medir parecidos, comparar contextos, transformar representaciones, combinar información.

Es como si hubiéramos pasado de un diccionario a un mapa [3].

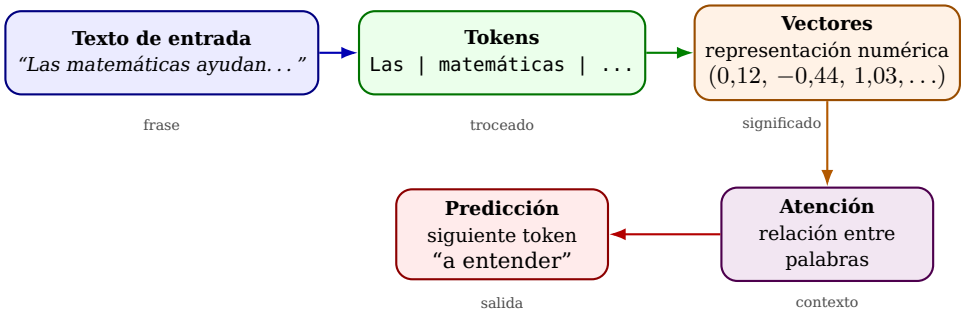

Figura 2: Esquema simplificado del funcionamiento de un modelo de lenguaje.

3. Geometría del significado: palabras como puntos

Pensar en palabras como puntos de un espacio puede parecer extraño al principio, pero la idea es potente. Si dos palabras aparecen en contextos parecidos, sus vectores tenderán a parecerse. Así, palabras como “profesor” y “docente”, o “gato” y “león”, tenderán a ocupar regiones próximas del espacio.

No significa que el modelo “entienda” el significado como lo entiende una persona. Pero sí captura regularidades estadísticas profundas: qué palabras aparecen cerca de cuáles, qué estructuras sintácticas se repiten, qué asociaciones son frecuentes en el lenguaje.

Podemos imaginar este espacio como una especie de universo geométrico del significado. No tiene dos ni tres dimensiones, como los dibujos del plano o del espacio que hacemos en clase, sino cientos o miles. Es imposible visualizarlo directamente, pero las nociones geométricas siguen funcionando. Hay distancias, direcciones, cercanías, agrupamientos.

Y entonces ocurre algo fascinante: relaciones lingüísticas empiezan a parecerse a relaciones geométricas.

Por ejemplo, en contextos parecidos pueden aparecer palabras como “rey” y “reina”, o “Madrid” y “España”, o “radio” y “circunferencia”. No porque el modelo conozca la monarquía, la geografía o la geometría como un ser humano, sino porque ha observado millones de textos y ha ajustado sus vectores para reflejar patrones de uso [5].

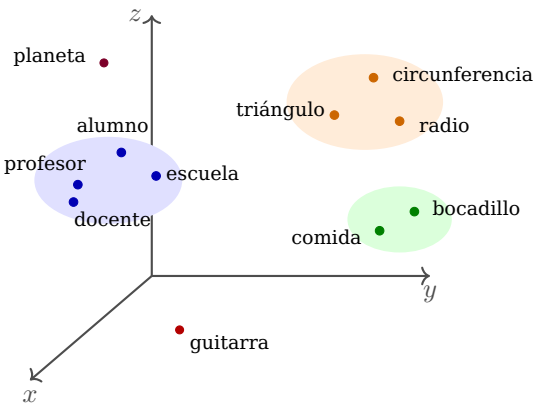

Figura 3: Representación esquemática de palabras como puntos en un espacio geométrico. Palabras que aparecen en contextos parecidos tienden a quedar cerca, mientras que palabras de ámbitos distintos aparecen más alejadas. La figura se dibuja en tres dimensiones por motivos visuales, aunque los modelos reales trabajan en espacios de muchas más dimensiones.

Esta representación también nos permite usar formulaciones matemáticas para llegar a un significado dentro del lenguaje, como podemos ver en la Figura 4.

Pero el lenguaje tiene una dificultad añadida. El significado de una palabra no siempre es fijo. La palabra “banco” no significa lo mismo en “me senté en un banco” que en “fui al banco a sacar dinero”. Así que no basta con asignar una posición geométrica inmóvil a cada palabra: hay que tener en cuenta el contexto.

Los LLMs modernos resuelven esto haciendo que la representación de cada token cambie según las palabras que lo rodean. Es decir, el vector de una palabra no depende solo de la propia palabra, sino también de la frase en la que aparece. El significado se vuelve dinámico.

Esto es una idea muy profunda: el lenguaje no funciona como una lista de piezas independientes, sino como una red de relaciones. El sentido emerge del contexto. Y lo que hace

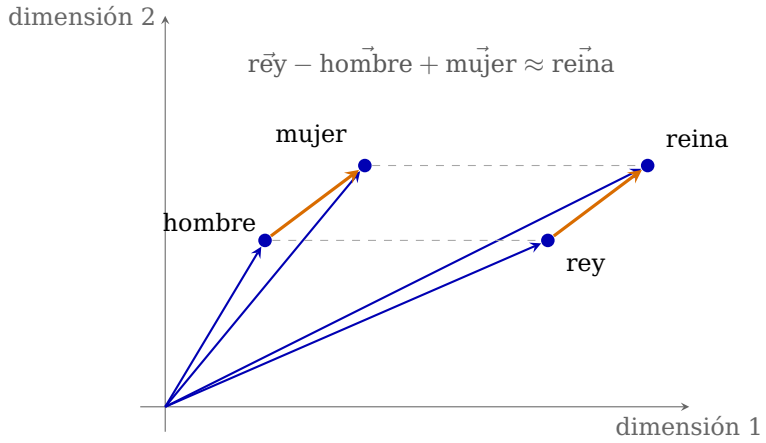

Figura 4: Esquema inspirado en las representaciones vectoriales del lenguaje: en un espacio semántico, ciertas relaciones entre palabras pueden representarse como desplazamientos entre vectores.

el modelo es reconstruir matemáticamente esa red, traduciendo relaciones lingüísticas en operaciones sobre vectores.

Ahora bien, una frase no es solo un conjunto de palabras relacionadas: también tiene orden. No da igual decir "el perro muerde al cartero" que "el cartero muerde al perro". Las palabras pueden ser las mismas, pero el significado cambia por completo.

De modo que el modelo necesita, además de vectores, una manera de representar el orden y decidir en cada momento qué partes de la frase son importantes para interpretar las demás.

4. El mecanismo de atención: cómo "mira" un LLM

La palabra "atención" suena psicológica, casi humana, pero en los LLMs tiene un significado matemático muy concreto.

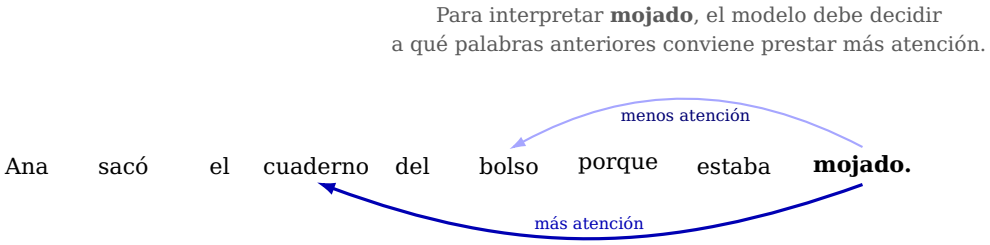

Figura 5: Esquema simplificado del mecanismo de atención. La palabra final *mojado* puede relacionarse con varias palabras anteriores, pero el modelo asigna pesos distintos a cada una. En este ejemplo, *cuaderno* recibe más atención que *bolso*.

Veamos la frase:

> "Ana sacó el cuaderno del bolso porque estaba mojado."

¿Qué estaba mojado?, ¿el cuaderno o el bolso? La frase es ambigua. En muchas otras frases, sin embargo, las relaciones son más claras. El modelo necesita aprender qué palabras deben influir en la interpretación de otras.

La atención hace justamente eso: permite que cada token “mire” al resto y decida cuáles le importan más.

No todos los elementos de una frase pesan igual en cada momento. Si estamos procesando un pronombre, quizá debamos fijarnos en un sustantivo anterior. Si aparece una negación, puede cambiar el sentido de lo que viene después. Si estamos al final de una pregunta, quizá debamos atender al principio para recuperar el tema.

En lugar de leer el texto de forma rígida, paso a paso, el modelo calcula pesos que indican cuánta importancia debe dar cada palabra a las demás. Matemáticamente, esos pesos se obtienen comparando vectores y transformándolos mediante funciones que los convierten en valores positivos que suman 1. Es decir, en algo parecido a un reparto de atención [7].

La idea esencial puede expresarse así: para interpretar una palabra, no basta con mirarla sola; hay que construir una versión enriquecida de ella a partir del contexto. Esto permite captar dependencias lejanas. En una frase larga, una palabra del principio puede ser decisiva para interpretar otra del final. Y el mecanismo de atención puede conectarlas directamente, sin necesidad de recorrer toda la cadena intermedia paso a paso.

Es como si, al leer, nuestra mente trazara hilos invisibles entre distintas partes de la oración. El LLM hace algo parecido, pero con matrices, productos escalares y normalizaciones.

Una de las maravillas de esta idea es que funciona en paralelo. En lugar de procesar cada palabra aisladamente, el modelo puede analizar muchas relaciones a la vez. Y no solo desde un punto de vista, sino desde varios. Por eso se habla de múltiples cabezas de atención: distintas “miradas” simultáneas que captan distintos tipos de relación.

Aunque uno no vea las matrices por dentro, el efecto es notable: el modelo aprende a decidir dónde conviene fijarse. A veces en una concordancia gramatical, a veces en una referencia lejana, a veces en una estructura repetida. . .

En el fondo, la atención es una idea preciosa porque traduce a una operación matemática precisa una intuición casi literaria: que el sentido depende de qué observamos y de cómo lo relacionamos.

5. La idea clave: predecir la siguiente palabra

Y ahora que tenemos las herramientas para que un ordenador entienda el texto podemos pasar a la tarea fundamental con la que se entrenan estos modelos. Aunque luego hagan muchas cosas distintas, su aprendizaje básico suele consistir en algo muy concreto: predecir el siguiente token.

Si el modelo recibe el comienzo de una frase como:

> “Las abejas construyen. . . ”

debe calcular qué tokens podrían venir a continuación y con qué probabilidad. Tal vez “panales”, “sus”, “colmenas” o “nidos”. Si el texto real sigue con “panales”, el sistema ajustará sus parámetros para aumentar la probabilidad de esa continuación en contextos similares.

Este proceso se repite millones y millones de veces con textos de todo tipo. Cada vez que el modelo falla, se corrige un poco. Cada vez que acierta, refuerza ciertos patrones. Con el tiempo va captando regularidades del lenguaje: concordancias, estructuras sintácticas, expresiones frecuentes, relaciones semánticas, formas de argumentar, estilos de escritura.

De nuevo, la idea parece simple. Pero si uno lo piensa un poco, predecir la siguiente palabra exige muchísimo conocimiento implícito.

En la frase:

> “Ayer dejé el helado al sol y cuando volví estaba. . . ”

la continuación más razonable es “derretido”. Para elegirla no basta con dominar la gramática. Hay que haber absorbido muchísimos ejemplos sobre objetos, calor, tiempo, acciones y consecuencias. El modelo no razona necesariamente como una persona, pero ha visto tantos textos que aprende correlaciones que, en la práctica, contienen mucha información sobre el mundo.

Predecir el siguiente token es como jugar constantemente a completar frases. Solo que en lugar de hacerlo con unas pocas oraciones, el modelo lo hace con bibliotecas enteras. Y en lugar de memorizar respuestas exactas, ajusta una inmensa red de números para mejorar en esa tarea.

Es importante notar que el sistema no escoge siempre la palabra más probable de forma mecánica. A menudo trabaja con una distribución de probabilidades. Es decir, no dice “la siguiente palabra es esta y ninguna otra”, sino “estas son plausibles, pero unas lo son más que otras”. Eso le da flexibilidad y permite generar textos variados.

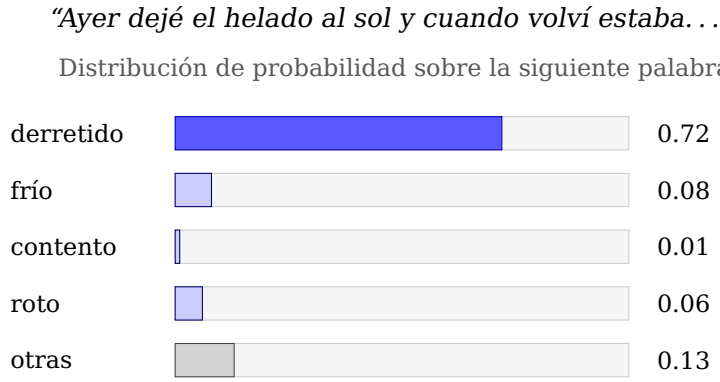

Figura 6: Ejemplo esquemático de predicción probabilística en un modelo de lenguaje. Ante una frase incompleta, el modelo no elige directamente una única palabra, sino que asigna distintas probabilidades a varias continuaciones posibles. La categoría *otras* agrupa el resto de palabras candidatas.

Desde el punto de vista matemático, aquí entra de lleno la probabilidad. El modelo no produce certezas, sino estimaciones. En cada paso calcula una especie de paisaje de posibilidades y avanza por él token a token.

6. Aprender ajustando millones de números

Llegados a este punto, ya tenemos varias piezas: tokens, vectores, contexto, atención y predicción. Pero queda la gran cuestión: ¿cómo aprende realmente el modelo?

La respuesta corta es: ajustando números.

Un LLM contiene una cantidad gigantesca de parámetros. Dependiendo del modelo, pueden ser millones o miles de millones. Esos parámetros son números reales que aparecen en todas las operaciones internas del sistema: en cómo se transforman los vectores, en cómo se comparan unas palabras con otras y en cómo se produce la salida final. Podemos imaginar cada parámetro como una pequeña rueda que se puede girar un poco. Aprender consiste, precisamente, en encontrar una posición adecuada para todas esas ruedas.

Para entender la idea, conviene mirar antes una red neuronal muchísimo más pequeña, casi de juguete. Supongamos que queremos construir un sistema muy simple que, a partir de dos características de una palabra o de una frase, produzca una puntuación final. Llamemos x_1 y x_2 a esas dos entradas. Una neurona muy sencilla podría calcular:

$$z = w_1 x_1 + w_2 x_2 + b,$$

donde w_1 y w_2 son pesos y b es un sesgo. Después aplicamos una función no lineal, por ejemplo:

$$a = \max(0; z),$$

que es una versión muy simple de las funciones que se usan en redes neuronales modernas. La idea es que la neurona combina las entradas, las pesa de distinta manera y produce una salida.

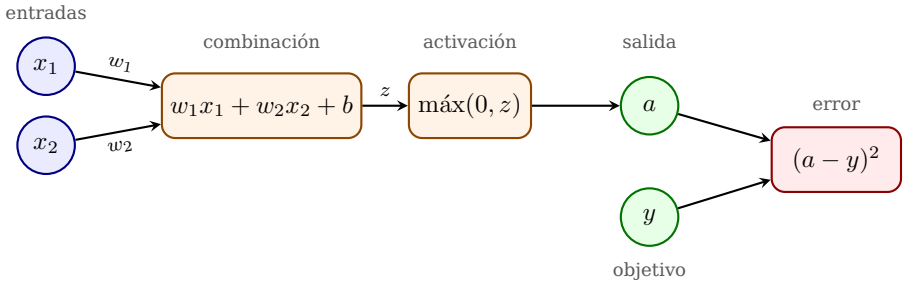

Figura 7: Una neurona artificial de juguete. Las entradas x_1 y x_2 se combinan con pesos w_1 y w_2, se añade un sesgo b, se aplica una función de activación y se obtiene una salida a, que después se compara con el valor deseado y.

Pongamos números concretos. Supongamos que:

$$x_1 = 2, \qquad x_2 = 1,$$

y que los parámetros iniciales son

$$w_1 = 0{,}5, \qquad w_2 = -0{,}3, \qquad b = 0{,}1.$$

Entonces la neurona calcula:

$$z = 0{,}5 \cdot 2 + (-0{,}3) \cdot 1 + 0{,}1 = 0{,}8,$$

y por tanto

$$a = \max(0; 0{,}8) = 0{,}8.$$

La salida de esta pequeña red sería $0{,}8$.

Ahora imaginemos que para ese ejemplo concreto nos gustaría que la salida correcta fuese 1. Entonces la red se ha quedado corta: ha dado $0{,}8$ en lugar de 1. Una forma muy sencilla de medir ese error es usar la diferencia al cuadrado:

$$L = (a - y)^2,$$

donde y es el valor correcto. En este caso,

$$L = (0{,}8 - 1)^2 = 0{,}04.$$

Ese número L se llama pérdida o error. Cuanto más pequeño sea, mejor está funcionando la red. Si la salida hubiera sido exactamente 1, entonces la pérdida sería 0.

La idea del aprendizaje consiste en cambiar un poco los parámetros w_1, w_2 y b para conseguir que la pérdida disminuya. Si al modificar uno de ellos el error baja, vamos en buena dirección; si sube, hemos empeorado. El problema es que en una red real hay demasiados parámetros como para ir probando a mano, por lo que usamos herramientas muy potentes de optimización.

En una red tan pequeña, todo este proceso puede parecer poca cosa. Pero ahora imaginemos miles de neuronas encadenadas en varias capas. Cada capa toma los números que salen de la anterior, los combina mediante matrices, añade sesgos, aplica funciones no lineales y entrega el resultado a la siguiente. Esquemáticamente, una capa puede escribirse como:

$$h^{(1)} = \sigma(W^{(1)}x + b^{(1)}),$$

y la siguiente como:

$$h^{(2)} = \sigma(W^{(2)}h^{(1)} + b^{(2)}).$$

Aquí x es el vector de entrada, $W^{(1)}$ y $W^{(2)}$ son matrices de pesos, $b^{(1)}$ y $b^{(2)}$ son vectores de sesgos y σ representa una función no lineal. Aunque la notación parezca compacta, la idea sigue siendo la misma que en la neurona de juguete: multiplicar, sumar y transformar.

Un LLM no es más que una versión gigantesca y sofisticada de este principio general. La entrada ya no son solo dos números, sino vectores muy grandes que representan tokens y contexto. En vez de una sola neurona hay muchísimas capas, y además aparecen mecanismos como la atención. Pero el corazón del aprendizaje no cambia: producir una salida, compararla con la deseada, medir el error y ajustar los parámetros para equivocarse menos la próxima vez.

Por eso decimos que el modelo aprende ajustando números. No memoriza reglas escritas a mano, ni recibe una explicación explícita de la gramática o del significado de cada palabra. Lo que hace es algo más mecánico y, a la vez, más sorprendente: modifica millones de pesos y sesgos hasta que ciertas configuraciones internas funcionan mejor que otras.

Es un proceso ciego y paciente. El modelo no “comprende” de repente una regla como quien tiene una revelación. No se detiene a pensar qué significa una frase ni descubre conscientemente la diferencia entre sujeto y predicado. Lo que hace es reajustar numéricamente su estructura interna hasta que ciertos patrones resultan más eficaces que otros.

Y, sin embargo, de ese mecanismo tan elemental emerge algo sorprendente. Para reducir bien el error, el modelo se ve obligado a captar regularidades profundas del lenguaje: qué palabras suelen aparecer juntas, qué estructuras sintácticas son frecuentes, cómo influyen unas partes de la frase sobre otras y qué continuaciones encajan mejor en cada contexto.

Lo asombroso es que, a partir de esta tarea tan básica, emergen capacidades muy sofisticadas. Un modelo grande, entrenado con muchísimos datos, acaba siendo capaz de resumir, traducir, responder preguntas o continuar textos en estilos muy distintos. No porque alguien le haya programado cada una de esas habilidades por separado, sino porque todas dependen, en parte, de captar regularidades profundas del lenguaje. El escalado de estos modelos mostró de forma muy clara que, al aumentar el tamaño del modelo y la cantidad de datos, aparecían habilidades cada vez más generales y flexibles [2].

Aquí aparece una idea muy matemática y también muy filosófica: la complejidad emergente. A veces, cuando un sistema tiene suficientes elementos e interacciones, aparecen comportamientos nuevos que no estaban presentes de forma obvia en las reglas microscópicas. En este caso, la regla microscópica es casi humilde: ajustar pesos y sesgos para reducir un error. Pero, repetida a gran escala sobre una enorme cantidad de texto, acaba produciendo comportamientos que, vistos desde fuera, parecen casi inteligentes.

Quizá esa sea una de las ideas más bellas de todo este asunto: que detrás de la aparente fluidez de un modelo de lenguaje no hay magia, sino una inmensa coreografía de matrices, vectores, funciones y gradientes intentando, una y otra vez, equivocarse un poco menos.

7. Por qué a veces aciertan... y a veces inventan

Después de todo lo anterior, quizá la pregunta más natural sea: si estos modelos son tan impresionantes, ¿por qué a veces se equivocan de forma tan extraña?, ¿por qué pueden escribir una explicación brillante y, unas líneas después, inventarse una referencia, una fecha o un dato falso, o dar una respuesta sin ninguna lógica?

La razón principal es que un LLM no está diseñado en primer lugar para distinguir la verdad de la falsedad, sino para producir texto plausible. Su objetivo básico es continuar secuencias de manera coherente con los patrones aprendidos. Y, aunque verdad y plausibilidad coinciden muchas veces, no son lo mismo. Un estudiante puede redactar una respuesta con gran seguridad y equivocarse; un modelo también. La diferencia es que el modelo puede hacerlo con una fluidez que lo hace especialmente convincente. Esta limitación ha dado lugar a debates importantes sobre los riesgos, sesgos y malentendidos asociados a los grandes modelos de lenguaje [1].

Por eso conviene entender bien qué son y qué no son estos sistemas. No son oráculos. No son bases de datos perfectas. No “piensan” como un ser humano. Son máquinas matemáticas extremadamente sofisticadas que han aprendido regularidades del lenguaje a una escala gigantesca.

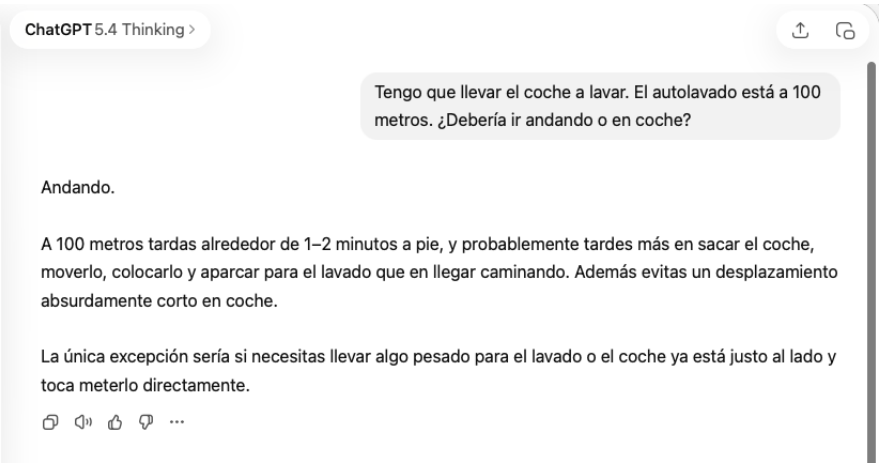

Figura 8: Ah, sí, lo normal, ir a lavar el coche andando

Y aun así, precisamente por eso, resultan fascinantes.

Nos enseñan que el lenguaje tiene más estructura de la que quizá imaginábamos. Nos muestran que la geometría y la probabilidad pueden llegar muy lejos cuando se aplican a textos masivos. Y nos recuerdan una vieja lección científica: que comprender un fenómeno no consiste en envolverlo en misterio, sino en encontrar el tipo adecuado de preguntas.

La próxima vez que un chatbot te responda con naturalidad, quizá merezca la pena mirar un poco más allá de la superficie. Detrás de esa respuesta no hay una voz atrapada dentro de una máquina, sino una inmensa arquitectura de vectores, probabilidades, productos matriciales y ajustes iterativos.

No hay magia. Pero, como ocurre tantas veces en matemáticas, la realidad acaba siendo más sorprendente que la magia.

Irina Arévalo es Doctora en Matemáticas y Doctora en Inteligencia Artificial. Actualmente es profesora en la Universidad Politécnica de Madrid, donde investiga en los fundamentos matemáticos de los algoritmos de Inteligencia Artificial, como por qué funcionan los sistemas de recomendación, cómo podemos compartir un modelo de IA, o cómo de justos son.

Bibliografía

[1] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[3] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2026. Online manuscript released January 6, 2026.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[6] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.