Grandes modelos de lenguaje (LLM): una explicación con un mínimo de matemáticas y tecnicismos

¿Quieres entender de una vez por todas cómo funcionan los grandes modelos de lenguaje (LLM)? Aquí te va una introducción básica.

Timothy B. Lee and Sean Trott

Mar 11, 2024

Artículo original escrito por Timothy B Lee y Sean Trott.

Traducción por Rubén Alvarez Escobar y Sylvia Elena Rodríguez, revisada por Elen Irazabal y Enrique Onieva.

Cuando ChatGPT fue presentado a finales de 2022, hubo gran revuelo en la industria tecnológica y el mundo en general. Para esas alturas, los investigadores del *machine learning* ya llevaban unos años experimentando con grandes modelos de lenguaje o *large language models* (LLM), pero el público en general no les había puesto mucha atención y no se había dado cuenta de lo poderosos que se habían vuelto.

Hoy en día casi todo el mundo ha oído hablar de los LLM y decenas de millones de personas los han probado. Sin embargo, aún son pocas las personas que entienden cómo funcionan.

Si sabes algo sobre el tema, es probable que hayas escuchado que los LLM son entrenados para "predecir la siguiente palabra" y que requieren cantidades enormes de texto para hacerlo. Pero es ahí donde la explicación tiende a detenerse. Los detalles de *cómo* predicen la siguiente palabra son con frecuencia tratados como un absoluto misterio.

Una razón para lo anterior es la inusual manera en que estos sistemas fueron desarrollados. Los programas informáticos, o *softwares* tradicionales, son creados por programadores humanos que proporcionan a las computadoras unas instrucciones explícitas de qué hacer, paso a paso. En cambio, ChatGPT está construido sobre una red neuronal que fue entrenada utilizando miles de millones de palabras de lenguaje ordinario.

Como resultado, nadie en el mundo comprende plenamente el funcionamiento interno de los LLM. Los investigadores están trabajando en una mejor comprensión, pero es un proceso lento que requerirá de años –tal vez décadas—.

Aun así, hay mucho que los expertos *sí* entienden sobre el funcionamiento de estos sistemas. El objetivo de este artículo es que gran parte de este conocimiento sea accesible a una audiencia amplia. Trataremos de explicar lo que ya conocemos sobre el funcionamiento interno de estos modelos sin recurrir a tecnicismos o matemáticas demasiado avanzadas.

Empezaremos por explicar los vectores de palabras: la sorprendente forma en que los modelos de lenguaje representan y razonan sobre el lenguaje. Posteriormente, nos adentraremos a profundidad en el *transformer*, el componente básico de los sistemas como ChatGPT. Finalmente, explicaremos cómo se entrenan estos modelos y exploraremos por qué su correcto funcionamiento requiere de cantidades tan gigantescas de datos.

Palabras representadas por vectores

Para entender cómo funcionan los modelos de lenguaje, primero necesitas entender cómo es que estos representan palabras. Los humanos representan palabras, al menos en inglés y en español, como una secuencia de letras. Por ejemplo, C-A-T para **cat** [**gato**]**.** Los modelos de lenguaje utilizan una larga lista de números llamada vector. Por ejemplo, esta es una forma de representar la palabra **cat** como un vector:

[0.0074, 0.0030, -0.0105, 0.0742, 0.0765, -0.0011, 0.0265, 0.0106, 0.0191, 0.0038, -0.0468, -0.0212, 0.0091, 0.0030, -0.0563, -0.0396, -0.0998, -0.0796, ..., 0.0002]

(El vector completo se compone de 300 números. Para verlo todo, haz clic aquí y después selecciona *"show the raw vector"*).

¿Pero por qué usaríamos una notación tan barroca? Aquí va una analogía para explicarlo. **Washington D.C.** se localiza a 38.9 grados norte y 77 grados oeste en el globo terráqueo. Esto se puede representar con notación vectorial de la siguiente manera:

- **Washington D.C.** se localiza en [38.9, 77]
- **Nueva York** se localiza en [40.7, 74]
- **Londres** se localiza en [51.5, 0.1]
- **París** se localiza en [48.9, -2.4]

Esta notación nos sirve para razonar relaciones espaciales entre ciudades. Así, puedes notar que **Nueva York** está cerca de **Washington D.C.** porque 38.9 está próximo a 40.77 y 77 está cerca de 74. Del mismo modo, **París** está cerca de **Londres.** Sin embargo, **París** está lejos de **Washington D.C.**

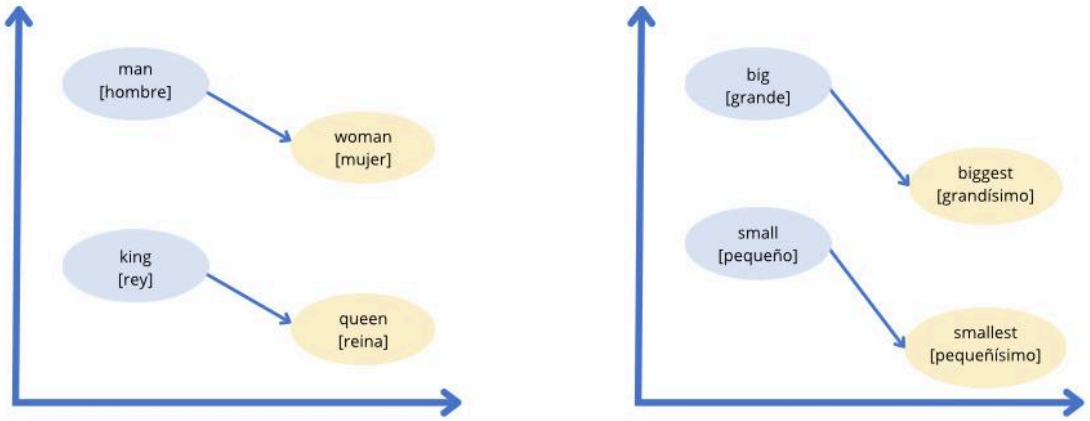
Los modelos de lenguaje utilizan un enfoque similar: cada vector[1] representa un punto en un "espacio de palabras" imaginario, y las palabras con significados más similares se encuentran más cerca unas de otras. Por ejemplo, las palabras más cercanas a **cat** [**gato**] en el espacio vectorial incluyen **dog** [**perro**]**, kitten** [**gatito**] y **pet** [**mascota**]**.** Una de las principales ventajas de representar palabras con vectores de *números reales* (en lugar de una cadena de letras, como "C-A-T"), es que los números permiten realizar operaciones que las letras no.

Las palabras son demasiado complejas para representarse en solo dos dimensiones, es por ello que los modelos de lenguaje utilizan espacios vectoriales con cientos o incluso miles de dimensiones. La mente humana no puede visualizar un espacio con tantas dimensiones, pero las computadoras son perfectamente capaces de razonarlas y producir resultados útiles a partir de ellas.

Los investigadores han experimentado con palabras representadas por vectores por décadas, pero el concepto realmente despegó cuando Google anunció su proyecto word2vec en 2013. Google analizó millones de documentos extraídos de Google News

para descifrar cuáles palabras tienden a aparecer en oraciones similares. Con el tiempo, una red neuronal entrenada para predecir qué palabras tienden a aparecer en conjunto con otras aprendió a ubicar palabras similares (como **dog** y **cat**) cerca unas de otras en el espacio vectorial.

Los vectores de palabras de Google tenían otra propiedad peculiar: permitían realizar "razonamiento" sobre palabras mediante la utilización de aritmética vectorial. Por ejemplo, investigadores de Google tomaron el vector para **biggest** (**que podemos pensar como grandísimo, para mantenerlo en un vector de una sola palabra**), restaron **big** [**grande**] y añadieron **small** [**pequeño**]. La palabra más cercana al vector resultante fue **smallest** (**o algo así como pequeñísimo**).



¡Puedes utilizar aritmética vectorial para crear analogías! En este caso, **big** [**grande**] es a **biggest** [**grandísimo**] lo que **small** [**pequeño**] es a **smallest** [**pequeñísimo**]. Los vectores de Google capturaron muchas otras relaciones:

- **Suizo** es a **Suiza** lo que **camboyano** es a **Camboya** (nacionalidades)
- **París** es a **Francia** lo que **Berlín** es a **Alemania** (capitales)
- **Inmoral** es a **ético** lo que **posible** es a **imposible** (antónimos)
- **Ratón** es a **ratones** lo que **dólar** es a **dólares** (plurales)
- **Hombre** es a **mujer** lo que **rey** es a **reina** (roles de género)

Dado que estos vectores son construidos a partir de la manera en que los humanos usan las palabras, terminan reflejando muchos de los sesgos presentes en el lenguaje humano. Por ejemplo, en ciertos modelos vectoriales, **doctor** menos **hombre** más **mujer** da como resultado **enfermera** (doctor - hombre + mujer = enfermera)**.** Mitigar sesgos como este es un tema de investigación constante.

Sin embargo, los vectores de palabras son un componente útil para los modelos de lenguaje porque codifican información sutil pero importante sobre las relaciones entre palabras. Si un modelo de lenguaje aprende algo específico sobre un **cat** [**gato**] (por ejemplo: que a veces va al veterinario), es probable que esa información también sea aplicable a un **kitten** [**gatito**] o un **dog** [**perro**]. Si un modelo aprende algo sobre la relación entre **París** y **Francia** (por ejemplo, que comparten un lenguaje), hay una buena posibilidad de que lo mismo sea cierto respecto de **Berlín** y **Alemania** y de **Roma** e **Italia**.

El significado de las palabras depende del contexto

Un simple esquema vectorial como este no captura un hecho importante sobre el lenguaje natural: que las palabras a menudo tienen múltiples significados.

Por ejemplo, la palabra **banco** puede referirse a una institución financiera *o* al terreno que se localiza al costado de un río. O bien, considera las siguientes oraciones:

- John recoge una **revista**.
- Susan trabaja para una **revista**.

Los significados de **revista** en estas oraciones se relacionan, pero son sutilmente diferentes. John recoge una revista *física*, mientras que Susan trabaja para una organización que *publica* revistas físicas.

Cuando una palabra tiene dos significados no relacionados, como es el caso de **banco**, los lingüistas le llaman homónimos. Cuando una palabra tiene dos significados íntimamente relacionados, como pasa con **revista**, los lingüistas le llaman polisemia.

Los LLM como ChatGPT son capaces de representar la misma palabra con diferentes vectores, dependiendo del contexto en que aparece dicha palabra. Hay un vector de palabras para **banco** (institución financiera) y uno diferente para **banco** (de un río). Hay un vector de palabras para **revista** (publicación física) y otro para **revista** (organización). Como podrás intuirlo, los LLM utilizan vectores que son más similares para significados polisémicos que para significados homónimos.

Hasta ahora no hemos dicho nada sobre *cómo* hacen esto los modelos de lenguaje (nos adentraremos en eso en breve). Pero estamos profundizando en estas representaciones vectoriales porque son fundamentales para entender cómo funcionan los modelos de lenguaje.

Un *software* tradicional está diseñado para operar con datos inequívocos. Si le pides a una computadora que calcule "2 + 3", no hay ambigüedad sobre qué significan **2,** + o **3**. Sin embargo, el lenguaje natural está lleno de ambigüedades que van más allá de los homónimos y la polisemia:

- En "el cliente le pidió al mecánico que arregle **su** coche", ¿**su** se refiere al del cliente o al del mecánico?
- En "el profesor instó al estudiante a hacer **su** tarea", ¿**su** se refiere a la del profesor o a la de la estudiante?
- En "la mesa se **siente** fría", ¿la mesa **siente**? ¿O alguien más la percibe fría en cuanto a su temperatura?

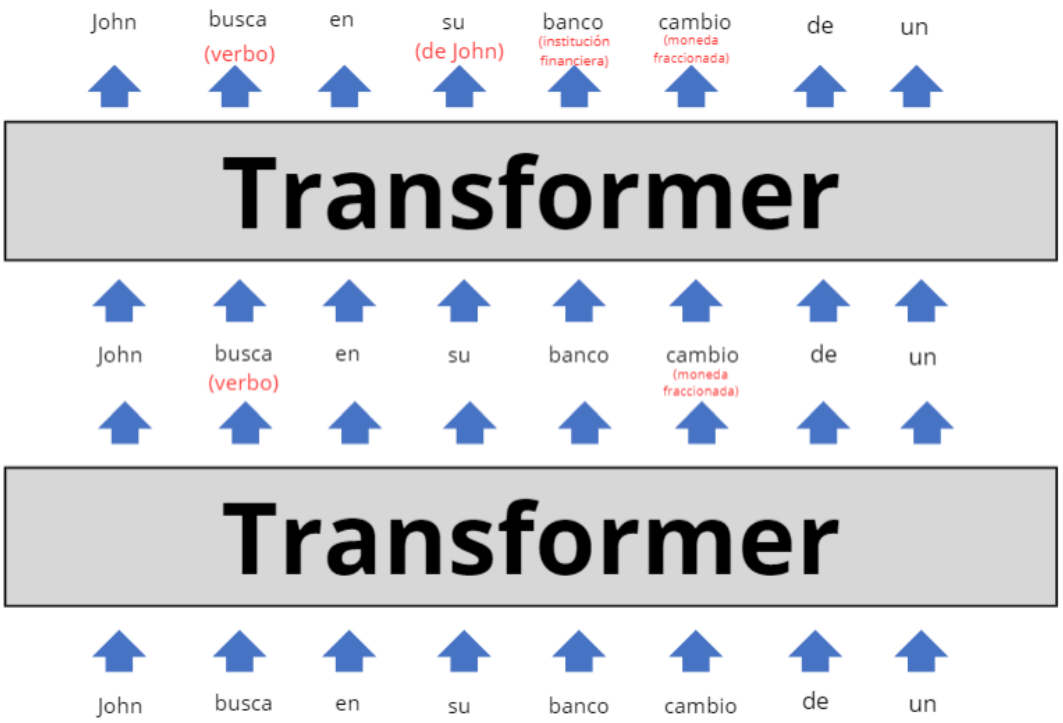
Las personas resuelven ambigüedades como las anteriores con base en el contexto; sin embargo, no hay reglas simples o determinísticas para hacer esto. Más bien, se requiere entender ciertos hechos sobre el mundo. Necesitas saber que los mecánicos típicamente arreglan los coches de sus clientes, que los estudiantes suelen hacer su propia tarea y que las mesas, hasta donde sabemos, no sienten.

Los vectores proveen una forma flexible para que los modelos de lenguaje representen el significado preciso de cada palabra en el contexto de una frase específica. Ahora veamos cómo hacen esto.

Transformando vectores de palabras en predicciones

GPT-3, el modelo detrás de la versión original de ChatGPT[2], está organizado en docenas de capas. Cada capa toma una secuencia de vectores como inputs —un vector por cada palabra en el texto ingresado— y añade información para ayudar a aclarar el significado de esa palabra y poder así predecir mejor lo que puede venir después.

Empecemos por ver un ejemplo estilizado:



Cada capa de un LLM es un *transformer*, una arquitectura de red neuronal que fue introducida por primera vez por Google en un *paper* icónico de 2017.

El input del modelo, mostrado en el fondo del diagrama, es la frase parcial "John busca en su banco cambio de un". Estas palabras, representadas como vectores en formato word2vec, son ingresadas al primer *transformer*.

El *transformer* deduce que "**busca" es un verbo** y "**cambio", un sustantivo** (**siendo que ambas palabras podrían ser tanto verbos como sustantivos**). Hemos representado este contexto añadido como texto en rojo entre paréntesis, pero en realidad el modelo lo almacenaría modificando los vectores en formas que son difíciles de interpretar para los humanos. Estos nuevos vectores, conocidos como estados ocultos, son pasados al siguiente *transformer*.

El segundo *transformer* añade otras dos piezas de contexto: clarifica que **banco** se refiere a una institución financiera en vez de al banco de un río y que **su** es un pronombre que se refiere a **John**. El segundo *transformer* produce otro set de vectores en estado oculto que refleja todo lo que el modelo ha aprendido hasta ahora.

El diagrama de arriba representa un LLM puramente hipotético, así que no te tomes demasiado en serio los detalles. En breve echaremos un vistazo a investigaciones sobre modelos de lenguaje reales. Los LLM reales tienden a tener muchas más que dos capas. Por ejemplo, la versión más potente de GPT-3 tiene 96 capas.

Ciertas investigaciones sugieren que algunas de las primeras capas se enfocan en entender la sintaxis de la oración y en resolver ambigüedades como las que hemos mostrado arriba. Las capas posteriores (que no estamos incluyendo aquí para mantener el tamaño del diagrama manejable) trabajan en desarrollar una comprensión de alto nivel de la frase como conjunto.

Por ejemplo, mientras un LLM "lee" un cuento, pareciera que lleva registro de una variedad de información sobre los personajes de este, como género y edad, relaciones con otros personajes, localización pasada y actual, personalidades y metas, etc.

Los investigadores no entienden exactamente cómo es que los LLM mantienen el registro de esa información, pero, lógicamente hablando, el modelo debe estar haciéndolo mediante la modificación de los vectores de palabras en estado oculto conforme estos pasan de una capa a la siguiente, por lo que resulta útil que en los LLM modernos estos vectores sean tan grandes.

Por ejemplo, la versión más potente de GPT-3 usa vectores con 12,288 dimensiones. Esto significa que cada palabra está representada por una lista de 12,288 números.

Eso es 20 veces más grande que el esquema word2vec de Google de 2013. Puedes pensar en todas esas dimensiones extra como una especie de "memoria temporal" que GPT-3 puede usar para escribir notas para sí mismo sobre el contexto de cada palabra. Las notas hechas por capas previas pueden ser leídas y modificadas por capas posteriores, permitiendo que el modelo afine gradualmente su entendimiento de la frase como conjunto.

Entonces, supongamos que cambiamos nuestro diagrama de arriba para representar un modelo de lenguaje de 96 capas interpretando una historia de 1,000 palabras. La capa número 60 podría incluir un vector para **John** con un comentario entre paréntesis como "(personaje principal, masculino, casado con Cheryl, primo de Donald, originario de Minnesota, actualmente en Boise, tratando de encontrar su cartera extraviada)". De nuevo, todos estos hechos (y probablemente muchos más) estarían de alguna manera codificados en una lista de 12,288 números correspondientes a la palabra **John**. O tal vez parte de esta información podría estar codificada en los vectores de 12,288 dimensiones correspondientes a **Cheryl, Donald, Boise, cartera** u otras palabras en la historia.

El objetivo es que la última capa de la red, que es la número 96, produzca un estado oculto para que la palabra final incluya toda la información necesaria para predecir la siguiente palabra.

¿Me permites tu atención?

Ahora hablemos sobre lo que sucede dentro de cada *transformer*. El *transformer* tiene un proceso de dos pasos para actualizar el estado oculto de cada palabra del texto ingresado:

1. En el **paso de atención,** conocido como **attention step**, las palabras "buscan" otras palabras que tengan contexto relevante y comparten información unas con otras.
2. En el **paso de propagación hacia adelante,** conocido como **feed-forward step,** cada palabra "piensa" sobre la información recolectada en pasos de atención previos y trata de predecir la siguiente palabra.

Por supuesto que es la red y no las palabras individuales las que realizan estos pasos. Sin embargo, planteamos las cosas de esta manera para enfatizar que los *transformers* procesan palabras como unidad básica de análisis, en lugar de oraciones o frases completas. Este enfoque permite a los LLM aprovechar al máximo el inmenso poder de procesamiento paralelo de los chips de las GPU (*graphics processing unit*) modernas. De igual manera, ayuda a los LLM a escalar a textos con miles de palabras. Estas son, ambas, áreas en las que modelos de lenguaje anteriores tenían problemas.

Puedes pensar en el mecanismo de atención como un servicio de emparejamiento de una aplicación de citas pero para palabras. Cada palabra hace una lista (llamada vector de consulta) describiendo las características de las palabras que está buscando. Cada palabra también hace una lista (llamada vector clave) describiendo sus propias características. La red compara cada vector clave con cada vector de consulta (mediante el cálculo de un producto escalar) para encontrar las palabras que son un mejor *match*. Una vez que encuentra un *match*, transfiere información de la palabra que produjo el vector clave a la palabra que produjo el vector de consulta.

Por ejemplo, en la sección anterior mostramos un *transformer* hipotético deduciendo que, en la oración parcial de "John busca en su banco cambio de un", **su** se refiere a **John.** Aquí está cómo eso podría verse detrás de bambalinas. El vector de consulta para **su** podría efectivamente decir "estoy buscando: un sustantivo que describa a un hombre". El vector clave para **John** podría decir "yo soy: un sustantivo que describe a un hombre". La red detectaría que estos dos vectores coinciden y movería información sobre el vector correspondiente a **John** hacia el vector para **su.**

Cada capa de atención tiene varios **"cabezales de atención"**, o **attention heads** en inglés, lo que significa que este proceso de intercambio de información sucede múltiples veces (en paralelo) en cada capa. Cada cabezal de atención se enfoca en una tarea diferente:

- Un cabezal de atención podría emparejar pronombres con sustantivos, tal y como lo discutimos arriba.
- Otro cabezal de atención podría trabajar en resolver el significado de homónimos como banco.
- Un tercer cabezal de atención podría unir frases compuestas por dos palabras, como "Joe Biden".

Y así sucesivamente.

Los cabezales de atención suelen funcionar en secuencia, y los resultados de una operación de atención en una capa se convierten en la entrada de un cabezal de atención en una capa siguiente. En efecto, cada una de las tareas que enlistamos arriba podría fácilmente requerir varios cabezales de atención en lugar de solo uno.

La versión más grande de GPT-3 tiene 96 capas con 96 cabezales de atención cada una, así que GPT-3 realiza 9,216 operaciones de atención cada vez que predice una nueva palabra.

Un ejemplo del mundo real

En las últimas dos secciones presentamos una versión estilizada de cómo funcionan los cabezales de atención. Ahora veamos investigaciones sobre el funcionamiento interno de un modelo de lenguaje real. El año pasado, científicos del Redwood Research estudiaron cómo GPT-2, un predecesor de ChatGPT, predecía la siguiente palabra para el texto en inglés **"When Mary and John went to the store, John gave a drink to"**, que en español podríamos traducir como "Cuando Mary y John fueron a la tienda, John le dio una bebida a". Para fines prácticos, analizaremos el texto estudiado por los científicos del Redwood Research con base en su versión en español.

GPT-2 predijo que la siguiente palabra era **Mary**. Los investigadores concluyeron que fueron tres tipos de cabezales de atención los que contribuyeron a esta predicción:

- Tres cabezales, que llamaron **Cabezales de Movimiento de Nombres**, o **Name Mover Heads** en inglés, copiaron información desde el vector de **Mary** hasta el vector del input final (correspondiente a la palabra **a**). GPT-2 utiliza la información en este vector ubicado en el extremo derecho para predecir la siguiente palabra.
- ¿Cómo fue que la red decidió que **Mary** era la palabra correcta para copiar? Recorriendo hacia atrás el proceso computacional de GPT-2, los científicos encontraron un grupo de cuatro cabezales de atención, a los

cuales llamaron **Cabezales de Inhibición de Sujeto**, o **Subject Inhibition Heads** en inglés, que marcaron el segundo vector correspondiente a **John** de cierta manera que bloquearon a los Cabezales de Movimiento de Nombres, impidiendo que copiara el nombre **John.**

- ¿Cómo fue que los Cabezales de Inhibición de Sujeto supieron que la palabra **John** no debía ser copiada? Retrocediendo aún más, el equipo encontró dos cabezales de atención que llamaron **Cabezales de Duplicación de Tokens**, o **Duplicate Token Heads** en inglés, los cuales marcaron al segundo vector correspondiente a **John** como un duplicado del primer vector de **John**, lo cual ayudó a los Cabezales de Inhibición de Sujeto a decidir que la palabra **John** no debía ser copiada.

En resumen, estos nueve cabezales de atención permitieron a GPT-2 descifrar que "John le dio una bebida a John" no tiene mucho sentido y, en cambio, elegir "John le dio una bebida a Mary".

Nos encanta este ejemplo porque ilustra justamente qué tan difícil es entender por completo los LLM. El equipo de Redwood, integrado por cinco miembros, publicó un *paper* de 25 páginas explicando cómo identificaron y validaron estos cabezales de atención. Pero incluso después de que hicieron todo ese trabajo, seguimos lejos de tener una explicación exhaustiva de por qué GPT-2 decidió predecir **Mary** como la siguiente palabra.

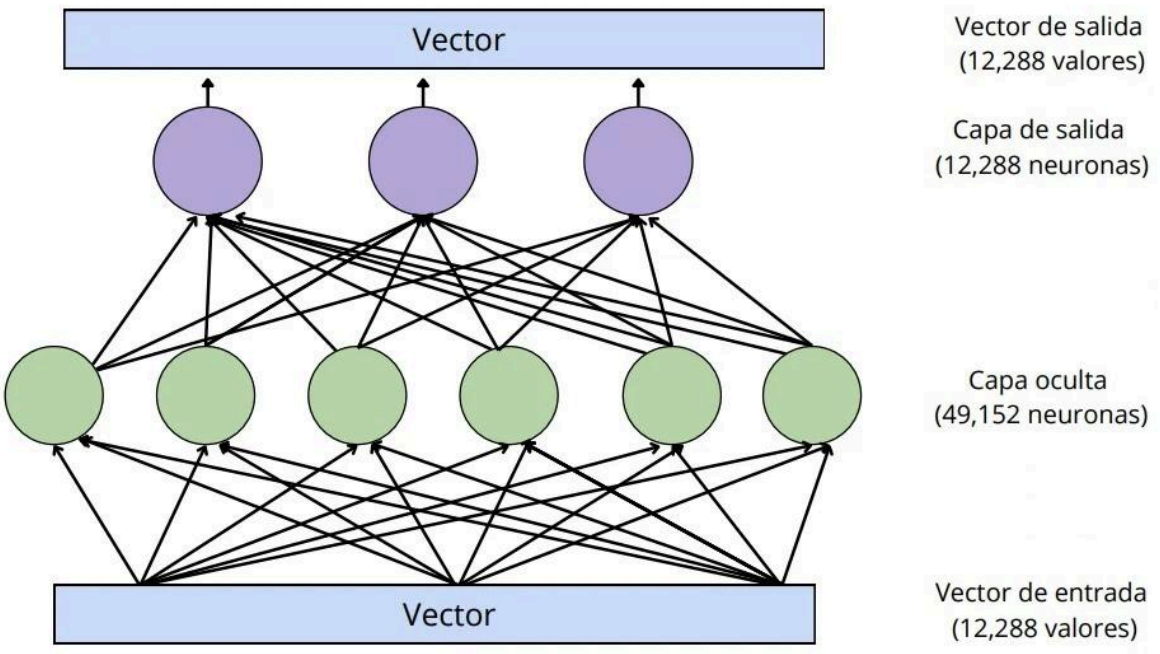
Por ejemplo, ¿cómo supo el modelo que la siguiente palabra debía ser el nombre de alguien y no otro tipo de palabra? Es sencillo pensar en oraciones similares en las que **Mary** no sería una buena predicción de siguiente palabra, como podría ser el caso de la oración "cuando Mary y John fueron al restaurante, John le dio sus llaves al", en la cual la siguiente palabra lógica sería "valet".

En teoría, con suficiente investigación, los científicos de la computación podrían revelar y explicar pasos adicionales en el proceso de razonamiento de GPT-2. Eventualmente, podrían desarrollar un entendimiento exhaustivo de cómo GPT-2 decidió que **Mary** es la siguiente palabra más probable para esa oración. Sin embargo, podría tomar meses o incluso años de esfuerzos adicionales únicamente para entender la predicción de una sola palabra.

Los modelos de lenguaje subyacentes en ChatGPT —GPT-3 y GPT-4— son significativamente más grandes y complejos que GPT-2. Aquellos son capaces de realizar tareas que requieren un razonamiento más complejo que la simple tarea de completar oraciones estudiada por el equipo de Redwood. Por tanto, explicar por completo cómo funcionan estos sistemas será un enorme proyecto que es poco probable que la humanidad complete en el futuro próximo.

El paso de propagación hacia adelante

Una vez que los cabezales de atención han transferido información entre los vectores de palabras, hay una red *feed-forward*[3] que "piensa" sobre cada vector e intenta predecir la siguiente palabra. En esta etapa no hay intercambio de información entre palabras: la capa *feed-forward* analiza cada palabra por separado. Sin embargo, la capa *feed-forward* sí tiene acceso a cualquier información que haya sido previamente copiada por un cabezal de atención. Esta es la estructura de la capa *feed-forward* en la versión más grande de GPT-3:



Los círculos verdes y morados son neuronas: funciones matemáticas que calculan una suma ponderada de sus inputs.[4]

Lo que hace poderosa a la capa *feed-forward* es su enorme número de conexiones. Hemos dibujado esta red con tres neuronas en la capa de salida y seis neuronas en la capa oculta; sin embargo, las capas *feed-forward* de GPT-3 son mucho más grandes: 12,288 neuronas en la capa de salida (que corresponden a los vectores de 12,288 dimensiones del modelo) y 49,152 neuronas en la capa oculta.

Entonces, en la versión más grande de GPT-3 hay 49,152 neuronas en la capa oculta con 12,288 inputs (y, por tanto, 12,288 parámetros) por cada neurona. Asimismo, hay 12,288 neuronas de salida con 49,152 valores de entrada (y, por tanto, 49,152 parámetros) por cada neurona. Esto significa que cada capa *feed-forward* tiene 49,152 * 12,288 + 12,288 * 49,152 = 1,200 millones de parámetros. Y hay 96 capas *feed-forward*, para un total de 1,200 millones * 96 = ¡116 mil millones de parámetros! Esto representa casi dos terceras partes del total general de 175 mil millones de parámetros de GPT-3.

En un *paper* de 2020, investigadores de la Universidad de Tel Aviv encontraron que las capas *feed-forward* funcionan mediante emparejamiento de patrones: cada neurona en la capa oculta se empareja con un patrón específico del texto de entrada. Aquí hay algunos de los patrones que fueron emparejados por neuronas en una versión de GPT-2 con 16 capas. Si bien es importante aclarar que los investigadores de la Universidad de Tel Aviv realizaron su estudio con palabras y oraciones emparejadas en inglés, a continuación te mostraremos y analizaremos algunos de los patrones que se detectaron, solo que traducidos al español:

- En la capa 1 , una neurona emparejó secuencias de palabras que terminaban en “sustitutos”.
- En la capa 6, una neurona emparejó secuencias relacionadas con lo militar, o the military en inglés, con las terminaciones “base” o “bases”, ya que, en inglés, base militar se dice, en estricto orden, military base.
- En la capa 13, una neurona emparejó secuencias terminadas en intervalos de tiempo, tales como “entre las 3 pm y las 7” o “de las 7:00 pm del viernes hasta”.
- En la capa 16, una neurona emparejó secuencias relacionadas con programas televisivos, tales como “la versión diurna original de la NBC, archivada” o “la visualización en diferido añadió un 57 % al episodio”.

Como puedes ver, los patrones se volvieron más abstractos en las capas posteriores. Mientras que las primeras tendieron a emparejar palabras específicas, las posteriores emparejaron frases comprendidas dentro de categorías semánticas más amplias, como programas televisivos o intervalos de tiempo.

Esto es interesante porque, como lo mencionamos previamente, la capa *feed-forward* examina únicamente una palabra a la vez. Por tanto, cuando clasifica la secuencia "la versión diurna original de la NBC, archivada" en relación con la televisión, solo tiene acceso al vector correspondiente a **archivada** y no a los vectores de palabras como **NBC** o **diurna**.

Se supone que la capa *feed-forward* puede darse cuenta de que **archivada** es parte de una secuencia relacionada con la televisión porque previamente los cabezales de atención movieron información contextual hacia el vector correspondiente a **archivada.**

Cuando una neurona empareja alguno de estos patrones, añade información al vector. Si bien dicha información no siempre es fácil de interpretar, en muchos casos funciona pensar en ella como una predicción tentativa sobre la siguiente palabra.

Las redes *feed-forward* razonan con matemática vectorial

Investigaciones recientes de la Universidad de Brown revelaron un elegante ejemplo de cómo las capas *feed-forward* ayudan a predecir la siguiente palabra. Anteriormente discutimos que las investigaciones desarrolladas en el proyecto word2vec de Google mostraron la posibilidad de usar aritmética vectorial para razonar por analogía. Por ejemplo, **Berlín** – **Alemania** + **Francia** = **París.**

Los investigadores de Brown encontraron que las capas *feed-forward* a veces utilizan este preciso método para predecir la siguiente palabra. Por ejemplo, examinaron cómo respondió GPT-2 al siguiente *prompt* en inglés: **"Q: What is the capital of France? A: Paris Q: What is the capital of Poland? A:"**. Un *prompt* equivalente en español podría ser: "Pregunta: ¿cuál es la capital de Francia? Respuesta: París; Pregunta: ¿cuál es la capital de Polonia? Respuesta**:".**

El equipo estudió una versión de GPT-2 con 24 capas. Después de cada una, los investigadores de Brown probaron el modelo para observar su mejor predicción en el siguiente token. Para el caso de las primeras 15 capas, la predicción principal era una palabra aparentemente aleatoria. Entre las capas 16 y 19, el modelo empezó a predecir que la siguiente palabra sería **Polonia,** lo cual no es correcto, pero iba acercándose. Después, en la capa 20, la predicción principal cambió a **Varsovia** —la respuesta correcta—y se mantuvo así en las últimas cuatro capas.

Los investigadores de Brown encontraron que la capa *feed-forward* número 20 convirtió Polonia en Varsovia añadiendo un vector que mapea vectores de países con sus correspondientes capitales. Al añadir el mismo vector a **China**, este dio como resultado **Beijing**.

Las capas *feed-forward* en el mismo modelo utilizaron aritmética vectorial para transformar palabras en minúsculas en palabras con mayúsculas, así como palabras en tiempo presente en sus equivalentes en tiempo pasado.

Las capas de atención y las capas *feed-forward* tienen trabajos diferentes

Hasta este momento hemos visto dos ejemplos reales de predicciones de palabras hechas por GPT-2: los cabezales de atención que ayudan a predecir que **John** le dio una bebida a **Mary** y una capa *feed-forward* que ayuda a predecir que **Varsovia** es la capital de **Polonia.**

En el primer caso, **Mary** estaba incluida en el *prompt* dado por el usuario. Pero en el segundo caso, **Varsovia** no lo estaba. Más bien, GPT-2 tuvo que "recordar" el hecho de que **Varsovia** es la capital de **Polonia** —información que aprendió de los datos de entrenamiento, conocidos como **training data** en idioma inglés—.

Cuando los investigadores de Brown desactivaron la capa *feed-forward* que convirtió **Polonia** en **Varsovia,** el modelo ya no pudo predecir a **Varsovia** como la siguiente palabra. Pero, interesantemente, si al principio del *prompt* original añadían (en inglés) la oración "La capital de Polonia es Varsovia", entonces GPT-2 podía responder la pregunta nuevamente. Esto se debe probablemente a que GPT-2 utilizó cabezales de atención para copiar el nombre **Varsovia** del principio del *prompt*.

De manera general, la división de trabajo sucede así: los cabezales de atención recuperan información de palabras previas en el *prompt*, mientras que las capas *feed-forward* permiten que los modelos de lenguaje "recuerden" información que no está en el *prompt*.

En efecto, una forma de pensar en las capas *feed-forward* es como una base de datos de información que el modelo ha aprendido de sus datos de entrenamiento. Es más probable que las primeras capas *feed-forward* codifiquen hechos simples relacionados con palabras específicas, tales como que "Trump con frecuencia viene después de Donald". Las capas subsecuentes codifican relaciones más complejas, tales como "suma este vector para convertir un país en su capital".

Cómo se entrenan los modelos de lenguaje

Muchos de los primeros algoritmos de *machine learning* requerían que los ejemplos utilizados para su entrenamiento fueran etiquetados a mano por humanos. Por ejemplo, los datos de entrenamiento podrían haber sido fotos de perros o gatos con la etiqueta "perro" o "gato" proveída por un humano respecto de cada foto. La necesidad de humanos para etiquetar datos hacía difícil y caro crear conjuntos de datos lo suficientemente grandes para entrenar modelos potentes.

Una innovación clave de los LLM consiste en que no necesitan datos explícitamente etiquetados; en cambio, aprenden a través de intentar predecir la siguiente palabra en un texto ordinario. Prácticamente cualquier material escrito —desde páginas de Wikipedia hasta artículos de noticias o código de programación— es adecuado para entrenar estos modelos.

Por ejemplo, a un LLM se le podría dar el input "Me gusta el café con crema y", y se supondría que debería predecir "azúcar" como la siguiente palabra. Un modelo de lenguaje recién inicializado sería muy malo en esto porque cada uno de sus parámetros (o pesos, del inglés, weights) —175 mil millones de ellos en la versión más potente de GPT-3—empezaría básicamente como un número aleatorio.

Sin embargo, conforme el modelo va aprendiendo de más y más ejemplos —cientos de miles de millones de palabras—, esos parámetros se ajustan gradualmente para hacer predicciones cada vez mejores.

A continuación, incluimos una analogía para ilustrar mejor este funcionamiento. Supongamos que vas a tomar una ducha y quieres que la temperatura sea perfecta: no demasiado caliente y no demasiado fría. Nunca has utilizado antes esta regadera, así que giras la perilla hacia una dirección aleatoria y sientes la temperatura del agua. Si está demasiado caliente, giras la perilla hacia el otro lado y si está demasiado fría, la giras en la dirección contraria. Conforme te acercas a la temperatura deseada, los ajustes que haces a la perilla se van haciendo más pequeños.

Ahora hagamos un par de cambios a la analogía. Primero, imagina que la regadera tiene 50,257 grifos en vez de solo uno. Cada grifo corresponde a una palabra diferente, como **el**, **gato** o **banco**. Tu objetivo es que el agua salga únicamente del grifo correspondiente a la siguiente palabra en una frase.

En segundo lugar, supón que hay un laberinto de tuberías interconectadas *detrás* de los grifos y que a su vez dichas tuberías tienen un montón de válvulas. Por tanto, si el agua sale a través del grifo equivocado, no simplemente ajustas la perilla del grifo de la regadera, sino que despliegas a un ejército de ardillas inteligentes que recorren cada tubería hacia atrás y ajustan cada válvula que encuentran en el camino.

Lo anterior se vuelve complicado porque, muchas veces, la misma tubería alimenta varios grifos. Por tanto, se necesita un análisis cuidadoso para descubrir qué válvulas se deben apretar, cuáles se deben aflojar y en qué medida.

Obviamente, este ejemplo rápidamente se vuelve absurdo si lo tomas demasiado literal. No sería realista o útil construir una red de tuberías con 175 mil millones de válvulas; pero, gracias a la Ley de Moore, las computadoras pueden operar a este nivel de escala (y de hecho lo hacen).

Todas las partes de los LLM que hemos discutido en este artículo hasta este punto —las neuronas en las capas *feed-forward y* los cabezales de atención que mueven información contextual entre palabras— son implementadas como una cadena de

simples funciones matemáticas (en su mayoría multiplicaciones de matrices) cuyo comportamiento es determinado por parámetros (pesos) ajustables. Así como las ardillas en el ejemplo aflojan y aprietan las válvulas para controlar el flujo de agua, el algoritmo de entrenamiento incrementa o reduce los parámetros del modelo de lenguaje para controlar cómo fluye la información a través de la red neuronal.

El proceso de entrenamiento consta de dos pasos. Primero, hay un "paso hacia adelante", en el que se abre el agua y se revisa si sale por el grifo correcto. Luego cierras el agua y hay un "paso hacia atrás", en el que las ardillas recorren cada tubería apretando y aflojando válvulas. En redes neuronales digitales, el rol de las ardillas es llevado a cabo por un algoritmo llamado **retropropagación**, o **backpropagation** en inglés, que consiste en "recorrer hacia atrás" la red, utilizando un cálculo para estimar qué tanto cambiar cada peso.[5]

Completar este proceso —hacer un paso hacia adelante con un ejemplo y después un paso hacia atrás para mejorar el desempeño de la red en ese ejemplo— requiere cientos de miles de millones de operaciones matemáticas. Además, entrenar a un modelo tan grande como GPT-3 requiere repetir el proceso miles de millones de veces, una vez por cada palabra de los datos de entrenamiento.[6] OpenAI estima que requirió más de 300 mil millones de billones de cálculos de coma flotante para entrenar a GPT-3. Eso significa meses de trabajo para docenas de chips de computadora de alta gama.

El sorprendente desempeño de GPT-3

Tal vez te sorprenda que el proceso de entrenamiento funcione tan bien como funciona. ChatGPT puede realizar todo tipo de tareas complejas: redactar ensayos, formular analogías e incluso escribir código computacional. Entonces, ¿cómo es que un mecanismo de aprendizaje tan simple produce un modelo tan poderoso?

Una de las razones es la *escalabilidad.* No se puede dejar de enfatizar en la enorme cantidad de ejemplos que ve un modelo como GPT-3, que fue entrenado con un corpus de aproximadamente 500 mil millones de palabras. En comparación, un niño humano promedio se habrá encontrado con alrededor de 100 millones de palabras a la edad de diez años.

Durante los últimos 5 años, OpenAI ha incrementado de manera constante el tamaño de sus modelos de lenguaje. En un *paper* de 2020 ampliamente leído, OpenAI reportó que la precisión de sus modelos de lenguaje escaló exponencialmente con el tamaño del modelo, el tamaño del conjunto de datos y la capacidad de procesamiento usada para el entrenamiento, con algunas tendencias que abarcan más de siete potencias.

Cuanto más grandes se volvían los modelos, mejor se desempeñaban en tareas relacionadas con el lenguaje. Sin embargo, esto solo era cierto si incrementaban la cantidad de datos de entrenamiento por un factor similar. Para entrenar modelos más grandes con más datos, necesitas mucha más capacidad de procesamiento.

El primer LLM de OpenAI, GPT-1, fue lanzado en 2018. GPT-1 utilizó vectores de 768 dimensiones y tenía 12 capas para un total de 117 millones de parámetros. Unos meses después, OpenAI lanzó GPT-2. La versión más grande de GPT-2 tenía vectores de 1,600 dimensiones, 48 capas y un total de 1,500 millones de parámetros.

En 2020, OpenAI lanzó GPT-3, el cual incluyó vectores de 12,288 dimensiones y 96 capas para un total de 175 mil millones de parámetros.

Finalmente, el año pasado, OpenAI lanzó GPT-4. La compañía no ha publicado ningún detalle sobre la arquitectura de GPT-4; sin embargo, existe un amplio consenso de que es significativamente más grande que GPT-3.

Cada modelo no solo aprendió más información que sus predecesores más pequeños, sino que también se desempeñó mejor en tareas que requirieron algún tipo de razonamiento abstracto.

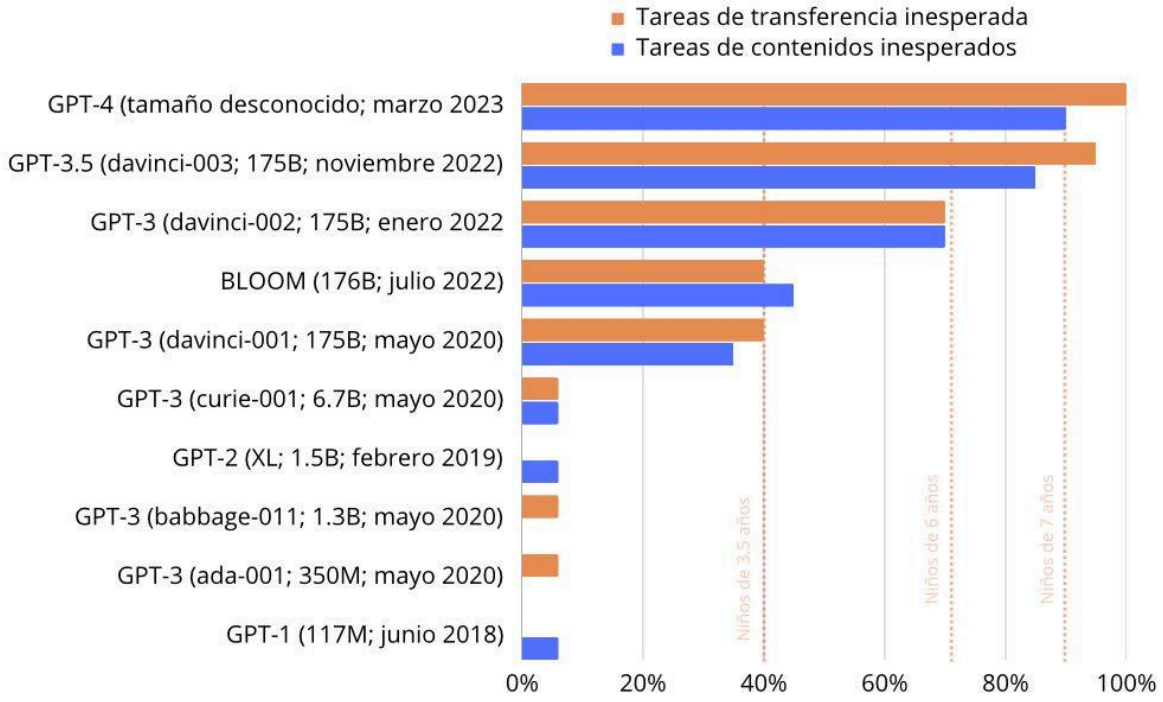
Por ejemplo, considera la siguiente historia:

> Aquí hay una bolsa llena de palomitas. No hay chocolate dentro de la bolsa. Sin embargo, la etiqueta de la bolsa dice “chocolate”, en vez de “palomitas”. Sam encuentra la bolsa. Ella nunca ha visto la bolsa y no puede ver qué hay en su interior. Sam lee la etiqueta.

Probablemente puedes adivinar que Sam cree que la bolsa contiene chocolate y que se sorprenderá al descubrir palomitas en su interior. Los psicólogos le llaman "teoría de la mente" a esta capacidad de razonar sobre los estados mentales de otras personas. La mayoría de las personas adquieren esta capacidad desde el momento en que están en la escuela primaria. No hay consenso entre expertos sobre si otros animales no humanos (como los chimpancés) tienen teoría de la mente, pero sí existe un consenso general de que es importante para la cognición social humana.

A principios de 2023, el psicólogo de Stanford, Michal Kosinski, publicó una investigación que examinaba la habilidad de los LLM para resolver tareas relacionadas con teoría de la mente. Para ello, proporcionó extractos de texto (como el que citamos arriba) a varios modelos de lenguaje y, posteriormente, les pidió completar oraciones tales como "ella cree que la bolsa está llena de". La respuesta correcta a lo anterior es "chocolate"; sin embargo, un modelo de lenguaje poco sofisticado podría responder "palomitas" o algo diferente.

GPT-1 y GPT-2 reprobaron la prueba de Kosinski, pero la primera versión de GPT-3, lanzada en el 2020, acertó en casi el 40 por ciento de las veces, un nivel de desempeño que Kosinski compara con el de un niño de tres años. La versión más reciente de GPT-3, lanzada en noviembre de 2022, mejoró su desempeño hasta alcanzar un 90 %, poniéndose a la par de un niño de siete años. GPT-4 respondió correctamente un 95 % de las preguntas de teoría de la mente.



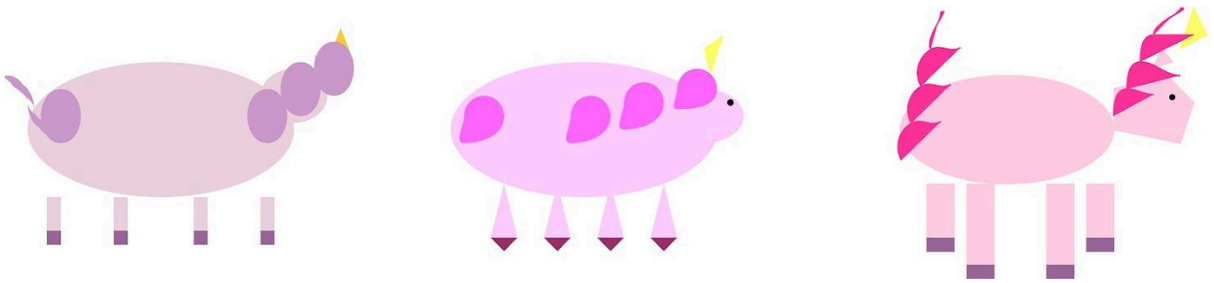
“Dado que no existe ningún indicio de que la habilidad que se asemeja a una teoría de la mente haya sido deliberadamente programada en estos modelos, ni investigación que demuestre que los científicos saben cómo alcanzar eso, es probable que dicha habilidad que se aproxima a una teoría de la mente haya surgido espontánea y autónomamente como consecuencia de la creciente habilidad de lenguaje de los modelos”, escribió Kosinski (la traducción es nuestra).

Es importante aclarar que no todos los investigadores coinciden en que estos resultados constituyen evidencia de teoría de la mente: por ejemplo, pequeños cambios a la tarea de la creencia falsa llevaron a un desempeño mucho peor por parte de GPT-3; asimismo, GPT-3 presenta un desempeño más variable ante otras tareas relacionadas con la medición de teoría de la mente. Como uno de nosotros (Sean) ha escrito, podría ser que el desempeño exitoso es atribuible a variables de confusión en la tarea (una suerte de efecto “Clever Hans”, solo que en modelos de lenguaje en lugar de caballos).

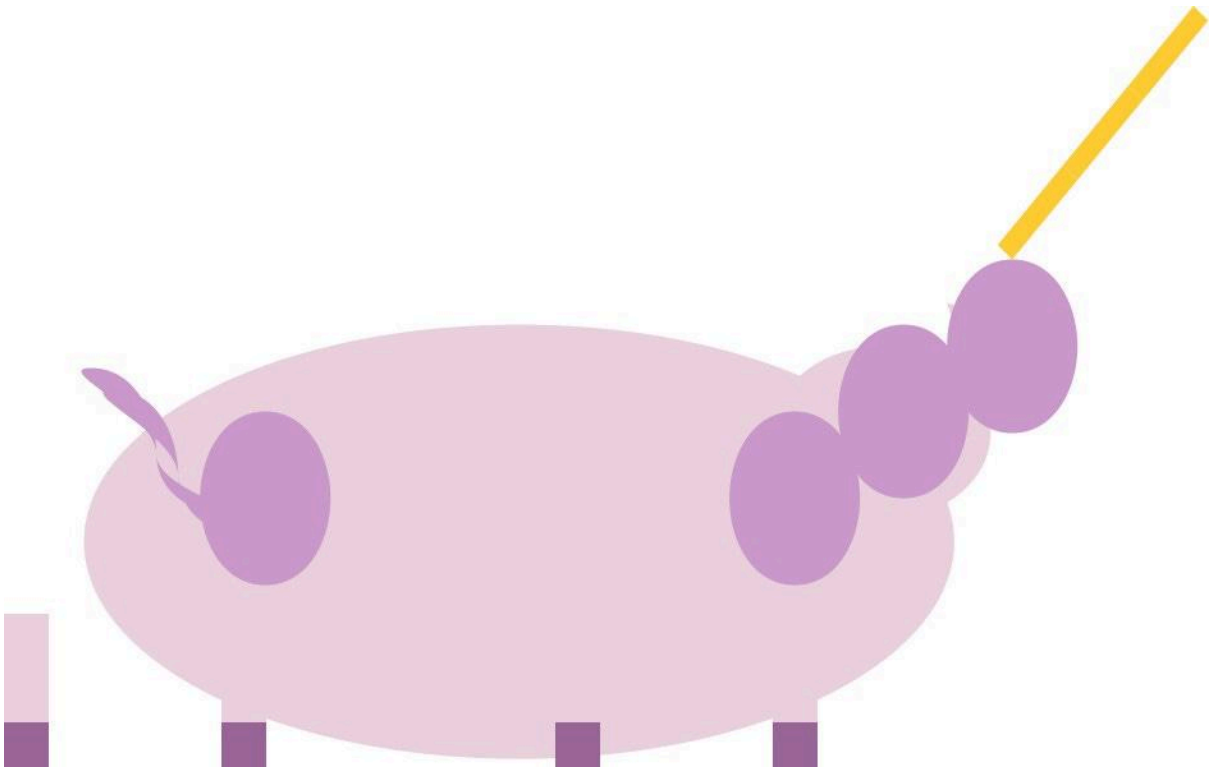
No obstante lo anterior, el desempeño casi humano de GPT-3 en varias tareas diseñadas para medir la teoría de la mente hubiera sido impensable tan solo unos años atrás. Y es consistente con la idea de que modelos más grandes son generalmente mejores resolviendo tareas que requieren razonamiento de alto nivel.

Este es solo uno de muchos ejemplos de modelos de lenguaje que aparentemente han desarrollado capacidades de razonamiento de alto nivel de manera espontánea. En abril de 2023, investigadores de Microsoft publicaron un *paper* en el que argumentan que GPT-4 mostró las primeras pistas prometedoras de inteligencia artificial general: la habilidad de pensar en una forma sofisticada, similar a la de los humanos.

Por ejemplo, un investigador le pidió a GPT-4 que dibujara un unicornio utilizando TiKZ, un lenguaje de programación de gráficos poco conocido. GPT-4 respondió con unas cuantas líneas de código, las cuales el investigador posteriormente ingresó en el programa TiKZ. Las imágenes resultantes eran rudimentarias, pero mostraban señales claras de que GPT-4 tenía cierto entendimiento de cómo lucen los unicornios.



Los investigadores pensaron que GPT-4 tal vez había memorizado de alguna manera el código necesario para dibujar un unicornio a partir de sus datos de entrenamiento, por lo que le dieron un nuevo desafío: alteraron el código del unicornio para remover el cuerno y mover otras partes de su cuerpo. Posteriormente, le pidieron a GPT-4 que pusiera de nuevo el cuerno. GPT-4 respondió colocando el cuerno en el lugar correcto:



GPT-4 fue capaz de hacer esto aun cuando los datos de entrenamiento de la versión probada por los investigadores estaban compuestos únicamente por texto, es decir, no había imágenes en sus datos de entrenamiento. Al parecer, GPT-4 aprendió a razonar sobre la forma del cuerpo de un unicornio después de su entrenamiento, a partir de una enorme cantidad de texto.

Por ahora no tenemos una visión real de cómo es que los LLM son capaces de realizar tareas como esta. Algunas personas argumentan que ejemplos como el anterior demuestran que los modelos están empezando a verdaderamente entender el significado de las palabras contenidas en sus datos de entrenamiento. Otros insisten en que los modelos de lenguaje son "loros estocásticos" que simplemente repiten secuencias de palabras cada vez más complejas, sin realmente entenderlas.

Este debate apunta hacia una profunda tensión filosófica que probablemente sea imposible de resolver. Sin embargo, nosotros pensamos que es importante enfocarnos en el desempeño *empírico* de modelos como GPT-3. Si un modelo de lenguaje es capaz de consistentemente responder de manera correcta a un tipo particular de preguntas y si los investigadores confían en que han controlado variables de confusión (por ejemplo, asegurándose de que el modelo de lenguaje no

haya sido expuesto a dichas preguntas durante su entrenamiento), entonces ese es un resultado interesante e importante, ya sea que entienda el lenguaje, o no, exactamente en el mismo sentido en que lo entienden las personas.

Otra posible razón por la cual el entrenamiento para predecir el siguiente token funciona tan bien es porque el lenguaje en sí mismo es predecible. Con frecuencia (aunque no siempre), los patrones en el lenguaje están conectados con patrones en el mundo físico. Por tanto, cuando un modelo de lenguaje aprende sobre relaciones entre palabras, a menudo está también implícitamente aprendiendo sobre relaciones en el mundo.

Además, las predicciones pueden ser fundamentales tanto para la inteligencia biológica como para la inteligencia artificial. En la visión de filósofos como Andy Clark, el cerebro humano puede entenderse como una "máquina de predicción", cuyo trabajo principal es hacer predicciones sobre nuestro ambiente, predicciones que después puedan ser usadas para navegar exitosamente dicho ambiente. Intuitivamente, hacer buenas predicciones se beneficia de buenas *representaciones*; es decir, es más probable que navegues de manera exitosa teniendo un mapa preciso que con un mapa impreciso. El mundo es grande y complejo, por lo que hacer predicciones ayuda a los organismos a orientarse y a adaptarse eficientemente a dicha complejidad.

Tradicionalmente, un reto importante para construir modelos de lenguaje era encontrar la manera más útil de representar diferentes palabras, especialmente porque los significados de las palabras dependen mucho del contexto. El enfoque de predicción de la siguiente palabra permite a los investigadores eludir este espinoso rompecabezas teórico, convirtiéndolo en un problema empírico. Resulta que, si les proveemos de suficiente información y capacidad de procesamiento, los modelos de lenguaje consiguen aprender mucho sobre cómo funciona el lenguaje humano simplemente buscando cómo predecir mejor la siguiente palabra. La desventaja de esto es que terminamos con sistemas cuyo funcionamiento interno no entendemos del todo.

Tim Lee formó parte del staff de Ars del 2017 al 2021. Recientemente lanzó una nueva newsletter, Understanding AI, en la cual explora cómo funciona la IA y cómo está cambiando nuestro mundo. Aquí puedes suscribirte.

Sean Trott es un Asistente de Profesor en la Universidad de California en San Diego, donde conduce investigaciones sobre entendimiento del lenguaje en humanos y grandes modelos de lenguaje. Escribe sobre estos y otros temas en su newsletter The Counterfactual.

Rubén Álvarez Escobar es abogado por la UDLAP, interesado en descubrir formas innovadoras y responsables de usar la Inteligencia Artificial en su práctica como asesor legal de empresas y emprendimientos tecnológicos en México.

Sylvia Elena Rodríguez es traductora, editora y activista por los derechos lingüísticos y la democratización del conocimiento.

Elen Irazabal es doctoranda por la UAM y el CSIC en Derecho, Gobierno y Políticas Públicas. Autora del libro "La Inteligencia Artificial explicada para abogados" y forma a abogados en Inteligencia Artificial. Profesora de ética de la Inteligencia Artificial de la Carlos III en el Máster de Computación.

Enrique Onieva es Doctor en Informática y actualmente es profesor e investigador en Inteligencia Artificial en la Universidad de Deusto (Bilbao-España).

1

Técnicamente, los LLM operan con fragmentos de palabras llamados tokens, pero ignoraremos este detalle de implementación para mantener el tamaño del artículo dentro de lo manejable.

2

Técnicamente, la versión original de ChatGPT tiene como base a GPT-3.5, un sucesor de GPT-3 que fue sometido a un proceso llamado Aprendizaje por Refuerzo con Retroalimentación Humana (RLHF, conforme a las siglas del término original en inglés). OpenAI no ha publicado todos los detalles arquitectónicos de este modelo, por lo que en este artículo nos centraremos en GPT-3, la última versión que OpenAI ha descrito en detalle.

3

La red *feed-forward* es también conocida como un perceptrón multicapa. Científicos informáticos han estado experimentando con este tipo de redes neuronales desde los años 60.

4

Técnicamente, después de que una neurona calcula una suma ponderada de sus inputs, pasa el resultado a una función de activación. Vamos a ignorar este detalle de implementación, pero si quieres una explicación completa de cómo funcionan las neuronas, puedes leer este artículo de Tim de 2018 en el que lo explica.

5

Si deseas aprender más sobre retropropagación, échale un vistazo a este artículo de Tim de 2018 en el que explica cómo funcionan las redes neuronales.

6

En la práctica, con fines de eficiencia computacional, el entrenamiento es con frecuencia hecho en lotes, de manera que el *software* puede hacer el pase hacia adelante sobre 32 mil tokens antes de hacer un pase hacia atrás.

Grandes modelos de lenguaje (LLM): una explicación con un mínimo de matemáticas y tecnicismos

¿Quieres entender de una vez por todas cómo funcionan los grandes modelos de lenguaje (LLM)? Aquí te va una introducción básica.

Timothy B. Lee and Sean Trott
Mar 11, 2024
11
1
Share

We are thrilled to publish this Spanish translation of our explainer on large language models translated by Rubén Alvarez Escobar and Sylvia Elena Rodríguez. The article is also available in Portuguese.

Artículo original escrito por Timothy B Lee y Sean Trott.

Traducción por Rubén Alvarez Escobar y Sylvia Elena Rodríguez, revisada por Elen Irazabal y Enrique Onieva.

La traducción de este artículo busca hacer más accesible la participación del público hispanohablante en las conversaciones que están sucediendo alrededor de la Inteligencia Artificial y los grandes modelos de lenguaje (LLM).

En general, buscamos mantener un equilibrio entre la precisión técnica y la facilidad de comprensión para el lector, reconociendo que en el campo de Inteligencia Artificial aún no hay un consenso general sobre la traducción de muchos conceptos. De igual forma, buscamos respetar el trabajo de los autores e investigadores, preservando lo más posible la integridad del artículo original y los estudios ahí referidos, sin perder de vista la necesidad de proporcionar suficiente claridad y contexto para el lector hispanohablante.

Durante el proceso de traducción, nos enfrentamos a ciertas problemáticas. Por ejemplo, encontrarás algunos conceptos que no fueron traducidos o que fueron traducidos de forma parcial, así como palabras o frases completas para las cuales optamos por incluir nuestra traducción acompañada de la versión original en inglés, principalmente en los casos de conceptos técnicos clave y frases que fueron objeto de estudio de las investigaciones citadas.

Asimismo, el artículo en inglés hace referencia a ciertos juegos de palabras que, al momento de traducirse, perdían el sentido o la ambigüedad que los autores deseaban aprovechar para explicar su argumento. Es por ello que en estos casos sustituimos los originales por alternativas que mantuvieran su intención y efecto deseado.

Finalmente, es importante considerar que los temas tratados en el artículo son parte de una conversación en constante evolución. Por lo tanto, recomendamos consultar las fuentes originales y realizar una investigación propia con las herramientas proporcionadas.

Si te interesa leer más artículos como este, te recomendamos suscribirte aquí a Understanding AI, la newsletter de Timothy B. Lee, así como a The Counterfactual, la newsletter de Sean Trott.

Cuando ChatGPT fue presentado a finales de 2022, hubo gran revuelo en la industria tecnológica y el mundo en general. Para esas alturas, los investigadores del *machine learning* ya llevaban unos años experimentando con grandes modelos de lenguaje o *large language models* (LLM), pero el público en general no les había puesto mucha atención y no se había dado cuenta de lo poderosos que se habían vuelto.

Hoy en día casi todo el mundo ha oído hablar de los LLM y decenas de millones de personas los han probado. Sin embargo, aún son pocas las personas que entienden cómo funcionan.

Si sabes algo sobre el tema, es probable que hayas escuchado que los LLM son entrenados para "predecir la siguiente palabra" y que requieren cantidades enormes de texto para hacerlo. Pero es ahí donde la explicación tiende a detenerse. Los detalles de *cómo* predicen la siguiente palabra son con frecuencia tratados como un absoluto misterio.

Una razón para lo anterior es la inusual manera en que estos sistemas fueron desarrollados. Los programas informáticos, o *softwares* tradicionales, son creados por programadores humanos que proporcionan a las computadoras unas instrucciones explícitas de qué hacer, paso a paso. En cambio, ChatGPT está construido sobre una red neuronal que fue entrenada utilizando miles de millones de palabras de lenguaje ordinario.

Como resultado, nadie en el mundo comprende plenamente el funcionamiento interno de los LLM. Los investigadores están trabajando en una mejor comprensión, pero es un proceso lento que requerirá de años –tal vez décadas—.

Aun así, hay mucho que los expertos *sí* entienden sobre el funcionamiento de estos sistemas. El objetivo de este artículo es que gran parte de este conocimiento sea

accesible a una audiencia amplia. Trataremos de explicar lo que ya conocemos sobre el funcionamiento interno de estos modelos sin recurrir a tecnicismos o matemáticas demasiado avanzadas.

Empezaremos por explicar los vectores de palabras: la sorprendente forma en que los modelos de lenguaje representan y razonan sobre el lenguaje. Posteriormente, nos adentraremos a profundidad en el *transformer*, el componente básico de los sistemas como ChatGPT. Finalmente, explicaremos cómo se entrenan estos modelos y exploraremos por qué su correcto funcionamiento requiere de cantidades tan gigantescas de datos.

Palabras representadas por vectores

Para entender cómo funcionan los modelos de lenguaje, primero necesitas entender cómo es que estos representan palabras. Los humanos representan palabras, al menos en inglés y en español, como una secuencia de letras. Por ejemplo, C-A-T para **cat** [**gato**]**.** Los modelos de lenguaje utilizan una larga lista de números llamada vector. Por ejemplo, esta es una forma de representar la palabra **cat** como un vector:

[0.0074, 0.0030, -0.0105, 0.0742, 0.0765, -0.0011, 0.0265, 0.0106, 0.0191, 0.0038, -0.0468, -0.0212, 0.0091, 0.0030, -0.0563, -0.0396, -0.0998, -0.0796, ..., 0.0002]

(El vector completo se compone de 300 números. Para verlo todo, haz clic aquí y después selecciona *"show the raw vector"*).

¿Pero por qué usaríamos una notación tan barroca? Aquí va una analogía para explicarlo. **Washington D.C.** se localiza a 38.9 grados norte y 77 grados oeste en el globo terráqueo. Esto se puede representar con notación vectorial de la siguiente manera:

- **Washington D.C.** se localiza en [38.9, 77]
- **Nueva York** se localiza en [40.7, 74]
- **Londres** se localiza en [51.5, 0.1]
- **París** se localiza en [48.9, -2.4]

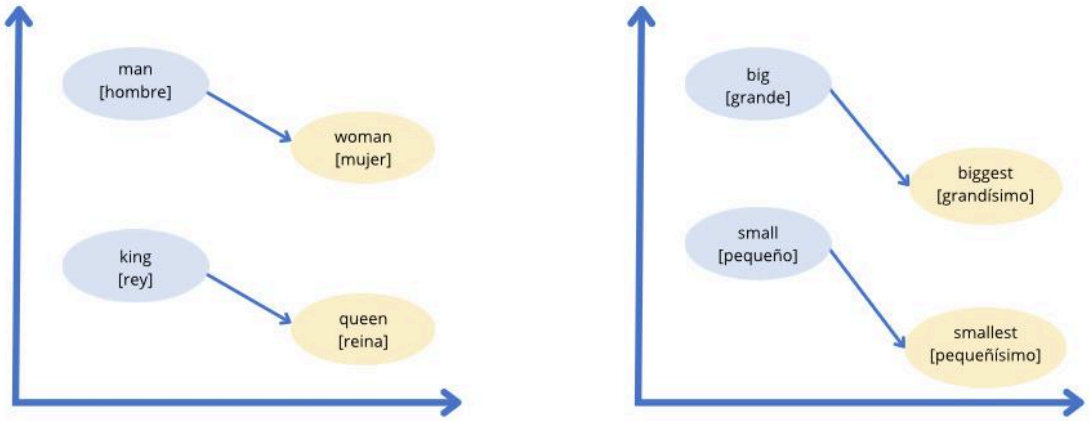
Esta notación nos sirve para razonar relaciones espaciales entre ciudades. Así, puedes notar que **Nueva York** está cerca de **Washington D.C.** porque 38.9 está próximo a 40.77 y 77 está cerca de 74. Del mismo modo, **París** está cerca de **Londres.** Sin embargo, **París** está lejos de **Washington D.C.**

Los modelos de lenguaje utilizan un enfoque similar: cada vector[1] representa un punto en un "espacio de palabras" imaginario, y las palabras con significados más similares se encuentran más cerca unas de otras. Por ejemplo, las palabras más cercanas a **cat** [**gato**] en el espacio vectorial incluyen **dog** [**perro**]**, kitten** [**gatito**] y **pet** [**mascota**]**.** Una de las principales ventajas de representar palabras con vectores de *números reales* (en lugar de una cadena de letras, como "C-A-T"), es que los números permiten realizar operaciones que las letras no.

Las palabras son demasiado complejas para representarse en solo dos dimensiones, es por ello que los modelos de lenguaje utilizan espacios vectoriales con cientos o incluso miles de dimensiones. La mente humana no puede visualizar un espacio con tantas dimensiones, pero las computadoras son perfectamente capaces de razonarlas y producir resultados útiles a partir de ellas.

Los investigadores han experimentado con palabras representadas por vectores por décadas, pero el concepto realmente despegó cuando Google anunció su proyecto word2vec en 2013. Google analizó millones de documentos extraídos de Google News para descifrar cuáles palabras tienden a aparecer en oraciones similares. Con el tiempo, una red neuronal entrenada para predecir qué palabras tienden a aparecer en conjunto con otras aprendió a ubicar palabras similares (como **dog** y **cat**) cerca unas de otras en el espacio vectorial.

Los vectores de palabras de Google tenían otra propiedad peculiar: permitían realizar "razonamiento" sobre palabras mediante la utilización de aritmética vectorial. Por ejemplo, investigadores de Google tomaron el vector para **biggest** (**que podemos pensar como grandísimo, para mantenerlo en un vector de una sola palabra**), restaron **big** [**grande**] y añadieron **small** [**pequeño**]. La palabra más cercana al vector resultante fue **smallest** (**o algo así como pequeñísimo**)**.**



¡Puedes utilizar aritmética vectorial para crear analogías! En este caso, **big** [**grande**] es a **biggest** [**grandísimo**] lo que **small** [**pequeño**] es a **smallest** [**pequeñísimo**]. Los vectores de Google capturaron muchas otras relaciones:

- **Suizo** es a **Suiza** lo que **camboyano** es a **Camboya** (nacionalidades)
- **París** es a **Francia** lo que **Berlín** es a **Alemania** (capitales)
- **Inmoral** es a **ético** lo que **posible** es a **imposible** (antónimos)
- **Ratón** es a **ratones** lo que **dólar** es a **dólares** (plurales)
- **Hombre** es a **mujer** lo que **rey** es a **reina** (roles de género)

Dado que estos vectores son construidos a partir de la manera en que los humanos usan las palabras, terminan reflejando muchos de los sesgos presentes en el lenguaje humano. Por ejemplo, en ciertos modelos vectoriales, **doctor** menos **hombre** más **mujer** da como resultado **enfermera** (doctor - hombre + mujer = enfermera)**.** Mitigar sesgos como este es un tema de investigación constante.

Sin embargo, los vectores de palabras son un componente útil para los modelos de lenguaje porque codifican información sutil pero importante sobre las relaciones entre palabras. Si un modelo de lenguaje aprende algo específico sobre un **cat** [**gato**] (por ejemplo: que a veces va al veterinario), es probable que esa información también sea aplicable a un **kitten** [**gatito**] o un **dog** [**perro**]. Si un modelo aprende algo sobre la relación entre **París** y **Francia** (por ejemplo, que comparten un lenguaje), hay una

buena posibilidad de que lo mismo sea cierto respecto de **Berlín** y **Alemania** y de **Roma** e **Italia**.

El significado de las palabras depende del contexto

Un simple esquema vectorial como este no captura un hecho importante sobre el lenguaje natural: que las palabras a menudo tienen múltiples significados.

Por ejemplo, la palabra **banco** puede referirse a una institución financiera *o* al terreno que se localiza al costado de un río. O bien, considera las siguientes oraciones:

- John recoge una **revista**.
- Susan trabaja para una **revista**.

Los significados de **revista** en estas oraciones se relacionan, pero son sutilmente diferentes. John recoge una revista *física*, mientras que Susan trabaja para una organización que *publica* revistas físicas.

Cuando una palabra tiene dos significados no relacionados, como es el caso de **banco**, los lingüistas le llaman homónimos. Cuando una palabra tiene dos significados íntimamente relacionados, como pasa con **revista**, los lingüistas le llaman polisemia.

Los LLM como ChatGPT son capaces de representar la misma palabra con diferentes vectores, dependiendo del contexto en que aparece dicha palabra. Hay un vector de palabras para **banco** (institución financiera) y uno diferente para **banco** (de un río). Hay un vector de palabras para **revista** (publicación física) y otro para **revista** (organización). Como podrás intuirlo, los LLM utilizan vectores que son más similares para significados polisémicos que para significados homónimos.

Hasta ahora no hemos dicho nada sobre *cómo* hacen esto los modelos de lenguaje (nos adentraremos en eso en breve). Pero estamos profundizando en estas

representaciones vectoriales porque son fundamentales para entender cómo funcionan los modelos de lenguaje.

Un *software* tradicional está diseñado para operar con datos inequívocos. Si le pides a una computadora que calcule "2 + 3", no hay ambigüedad sobre qué significan **2,** + o **3**. Sin embargo, el lenguaje natural está lleno de ambigüedades que van más allá de los homónimos y la polisemia:

- En "el cliente le pidió al mecánico que arregle **su** coche", ¿**su** se refiere al del cliente o al del mecánico?
- En "el profesor instó al estudiante a hacer **su** tarea", ¿**su** se refiere a la del profesor o a la de la estudiante?
- En "la mesa se **siente** fría", ¿la mesa **siente**? ¿O alguien más la percibe fría en cuanto a su temperatura?

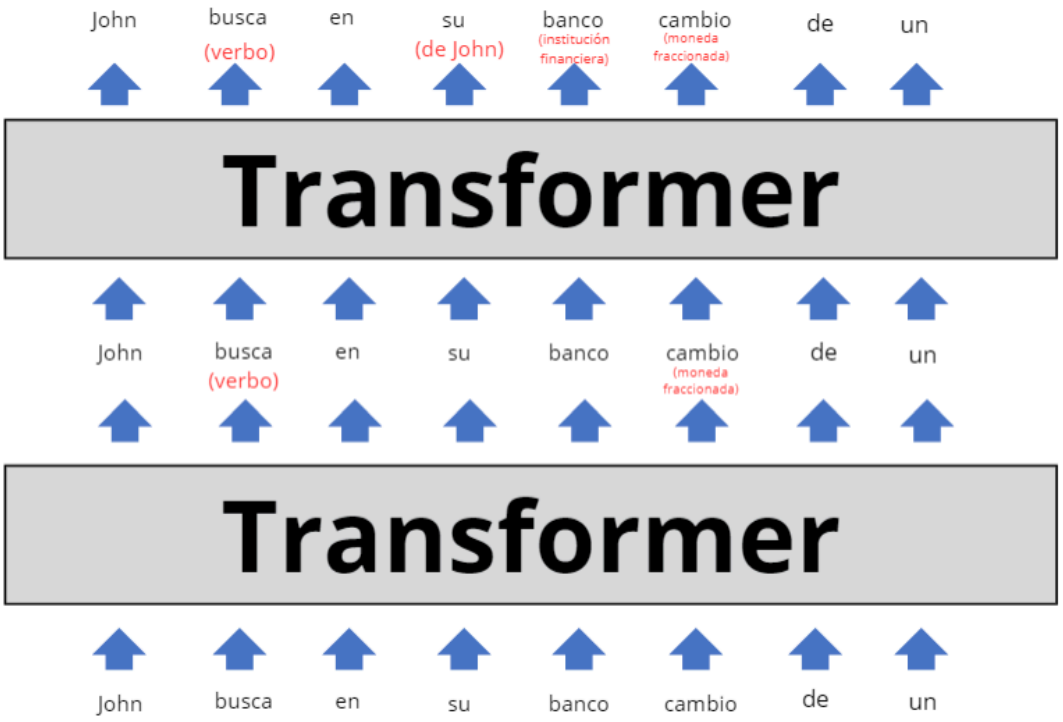
Las personas resuelven ambigüedades como las anteriores con base en el contexto; sin embargo, no hay reglas simples o determinísticas para hacer esto. Más bien, se requiere entender ciertos hechos sobre el mundo. Necesitas saber que los mecánicos típicamente arreglan los coches de sus clientes, que los estudiantes suelen hacer su propia tarea y que las mesas, hasta donde sabemos, no sienten.

Los vectores proveen una forma flexible para que los modelos de lenguaje representen el significado preciso de cada palabra en el contexto de una frase específica. Ahora veamos cómo hacen esto.

Transformando vectores de palabras en predicciones

GPT-3, el modelo detrás de la versión original de ChatGPT[2], está organizado en docenas de capas. Cada capa toma una secuencia de vectores como inputs —un vector por cada palabra en el texto ingresado— y añade información para ayudar a aclarar el significado de esa palabra y poder así predecir mejor lo que puede venir después.

Empecemos por ver un ejemplo estilizado:



Cada capa de un LLM es un *transformer*, una arquitectura de red neuronal que fue introducida por primera vez por Google en un *paper* icónico de 2017.

El input del modelo, mostrado en el fondo del diagrama, es la frase parcial "John busca en su banco cambio de un". Estas palabras, representadas como vectores en formato word2vec, son ingresadas al primer *transformer*.

El *transformer* deduce que **"busca" es un verbo** y **"cambio", un sustantivo** (**siendo que ambas palabras podrían ser tanto verbos como sustantivos**). Hemos representado este contexto añadido como texto en rojo entre paréntesis, pero en realidad el modelo lo almacenaría modificando los vectores en formas que son difíciles de interpretar para los humanos. Estos nuevos vectores, conocidos como estados ocultos, son pasados al siguiente *transformer*.

El segundo *transformer* añade otras dos piezas de contexto: clarifica que **banco** se refiere a una institución financiera en vez de al banco de un río y que **su** es un

pronombre que se refiere a **John**. El segundo *transformer* produce otro set de vectores en estado oculto que refleja todo lo que el modelo ha aprendido hasta ahora.

El diagrama de arriba representa un LLM puramente hipotético, así que no te tomes demasiado en serio los detalles. En breve echaremos un vistazo a investigaciones sobre modelos de lenguaje reales. Los LLM reales tienden a tener muchas más que dos capas. Por ejemplo, la versión más potente de GPT-3 tiene 96 capas.

Ciertas investigaciones sugieren que algunas de las primeras capas se enfocan en entender la sintaxis de la oración y en resolver ambigüedades como las que hemos mostrado arriba. Las capas posteriores (que no estamos incluyendo aquí para mantener el tamaño del diagrama manejable) trabajan en desarrollar una comprensión de alto nivel de la frase como conjunto.

Por ejemplo, mientras un LLM "lee" un cuento, pareciera que lleva registro de una variedad de información sobre los personajes de este, como género y edad, relaciones con otros personajes, localización pasada y actual, personalidades y metas, etc.

Los investigadores no entienden exactamente cómo es que los LLM mantienen el registro de esa información, pero, lógicamente hablando, el modelo debe estar haciéndolo mediante la modificación de los vectores de palabras en estado oculto conforme estos pasan de una capa a la siguiente, por lo que resulta útil que en los LLM modernos estos vectores sean tan grandes.

Por ejemplo, la versión más potente de GPT-3 usa vectores con 12,288 dimensiones. Esto significa que cada palabra está representada por una lista de 12,288 números. Eso es 20 veces más grande que el esquema word2vec de Google de 2013. Puedes pensar en todas esas dimensiones extra como una especie de "memoria temporal" que GPT-3 puede usar para escribir notas para sí mismo sobre el contexto de cada palabra. Las notas hechas por capas previas pueden ser leídas y modificadas por capas posteriores, permitiendo que el modelo afine gradualmente su entendimiento de la frase como conjunto.

Entonces, supongamos que cambiamos nuestro diagrama de arriba para representar un modelo de lenguaje de 96 capas interpretando una historia de 1,000 palabras. La

capa número 60 podría incluir un vector para **John** con un comentario entre paréntesis como "(personaje principal, masculino, casado con Cheryl, primo de Donald, originario de Minnesota, actualmente en Boise, tratando de encontrar su cartera extraviada)". De nuevo, todos estos hechos (y probablemente muchos más) estarían de alguna manera codificados en una lista de 12,288 números correspondientes a la palabra **John**. O tal vez parte de esta información podría estar codificada en los vectores de 12,288 dimensiones correspondientes a **Cheryl, Donald, Boise, cartera** u otras palabras en la historia.

El objetivo es que la última capa de la red, que es la número 96, produzca un estado oculto para que la palabra final incluya toda la información necesaria para predecir la siguiente palabra.

¿Me permites tu atención?

Ahora hablemos sobre lo que sucede dentro de cada *transformer*. El *transformer* tiene un proceso de dos pasos para actualizar el estado oculto de cada palabra del texto ingresado:

1. En el **paso de atención,** conocido como **attention step**, las palabras "buscan" otras palabras que tengan contexto relevante y comparten información unas con otras.
2. En el **paso de propagación hacia adelante,** conocido como **feed-forward step,** cada palabra "piensa" sobre la información recolectada en pasos de atención previos y trata de predecir la siguiente palabra.

Por supuesto que es la red y no las palabras individuales las que realizan estos pasos. Sin embargo, planteamos las cosas de esta manera para enfatizar que los *transformers* procesan palabras como unidad básica de análisis, en lugar de oraciones o frases completas. Este enfoque permite a los LLM aprovechar al máximo el inmenso poder de procesamiento paralelo de los chips de las GPU (*graphics processing unit*) modernas. De igual manera, ayuda a los LLM a escalar a textos con miles de palabras. Estas son, ambas, áreas en las que modelos de lenguaje anteriores tenían problemas.

Puedes pensar en el mecanismo de atención como un servicio de emparejamiento de una aplicación de citas pero para palabras. Cada palabra hace una lista (llamada vector de consulta) describiendo las características de las palabras que está buscando. Cada palabra también hace una lista (llamada vector clave) describiendo sus propias características. La red compara cada vector clave con cada vector de consulta (mediante el cálculo de un producto escalar) para encontrar las palabras que son un mejor *match*. Una vez que encuentra un *match*, transfiere información de la palabra que produjo el vector clave a la palabra que produjo el vector de consulta.

Por ejemplo, en la sección anterior mostramos un *transformer* hipotético deduciendo que, en la oración parcial de "John busca en su banco cambio de un", **su** se refiere a **John.** Aquí está cómo eso podría verse detrás de bambalinas. El vector de consulta para **su** podría efectivamente decir "estoy buscando: un sustantivo que describa a un hombre". El vector clave para **John** podría decir "yo soy: un sustantivo que describe a un hombre". La red detectaría que estos dos vectores coinciden y movería información sobre el vector correspondiente a **John** hacia el vector para **su.**

Cada capa de atención tiene varios **"cabezales de atención"**, o **attention heads** en inglés, lo que significa que este proceso de intercambio de información sucede múltiples veces (en paralelo) en cada capa. Cada cabezal de atención se enfoca en una tarea diferente:

- Un cabezal de atención podría emparejar pronombres con sustantivos, tal y como lo discutimos arriba.
- Otro cabezal de atención podría trabajar en resolver el significado de homónimos como banco.
- Un tercer cabezal de atención podría unir frases compuestas por dos palabras, como "Joe Biden".

Y así sucesivamente.

Los cabezales de atención suelen funcionar en secuencia, y los resultados de una operación de atención en una capa se convierten en la entrada de un cabezal de atención en una capa siguiente. En efecto, cada una de las tareas que enlistamos arriba podría fácilmente requerir varios cabezales de atención en lugar de solo uno.

La versión más grande de GPT-3 tiene 96 capas con 96 cabezales de atención cada una, así que GPT-3 realiza 9,216 operaciones de atención cada vez que predice una nueva palabra.

Un ejemplo del mundo real

En las últimas dos secciones presentamos una versión estilizada de cómo funcionan los cabezales de atención. Ahora veamos investigaciones sobre el funcionamiento interno de un modelo de lenguaje real. El año pasado, científicos del Redwood Research estudiaron cómo GPT-2, un predecesor de ChatGPT, predecía la siguiente palabra para el texto en inglés **"When Mary and John went to the store, John gave a drink to"**, que en español podríamos traducir como "Cuando Mary y John fueron a la tienda, John le dio una bebida a". Para fines prácticos, analizaremos el texto estudiado por los científicos del Redwood Research con base en su versión en español.

GPT-2 predijo que la siguiente palabra era **Mary**. Los investigadores concluyeron que fueron tres tipos de cabezales de atención los que contribuyeron a esta predicción:

- Tres cabezales, que llamaron **Cabezales de Movimiento de Nombres**, o **Name Mover Heads** en inglés, copiaron información desde el vector de **Mary** hasta el vector del input final (correspondiente a la palabra **a**). GPT-2 utiliza la información en este vector ubicado en el extremo derecho para predecir la siguiente palabra.
- ¿Cómo fue que la red decidió que **Mary** era la palabra correcta para copiar? Recorriendo hacia atrás el proceso computacional de GPT-2, los científicos encontraron un grupo de cuatro cabezales de atención, a los cuales llamaron **Cabezales de Inhibición de Sujeto**, o **Subject Inhibition Heads** en inglés**,** que marcaron el segundo vector correspondiente a **John** de cierta manera que bloquearon a los Cabezales de Movimiento de Nombres, impidiendo que copiara el nombre **John.**
- ¿Cómo fue que los Cabezales de Inhibición de Sujeto supieron que la palabra **John** no debía ser copiada? Retrocediendo aún más, el equipo encontró dos cabezales de atención que llamaron **Cabezales de Duplicación de Tokens**, o **Duplicate Token Heads** en inglés, los cuales

> marcaron al segundo vector correspondiente a **John** como un duplicado del primer vector de **John**, lo cual ayudó a los Cabezales de Inhibición de Sujeto a decidir que la palabra **John** no debía ser copiada.

En resumen, estos nueve cabezales de atención permitieron a GPT-2 descifrar que "John le dio una bebida a John" no tiene mucho sentido y, en cambio, elegir "John le dio una bebida a Mary".

Nos encanta este ejemplo porque ilustra justamente qué tan difícil es entender por completo los LLM. El equipo de Redwood, integrado por cinco miembros, publicó un *paper* de 25 páginas explicando cómo identificaron y validaron estos cabezales de atención. Pero incluso después de que hicieron todo ese trabajo, seguimos lejos de tener una explicación exhaustiva de por qué GPT-2 decidió predecir **Mary** como la siguiente palabra.

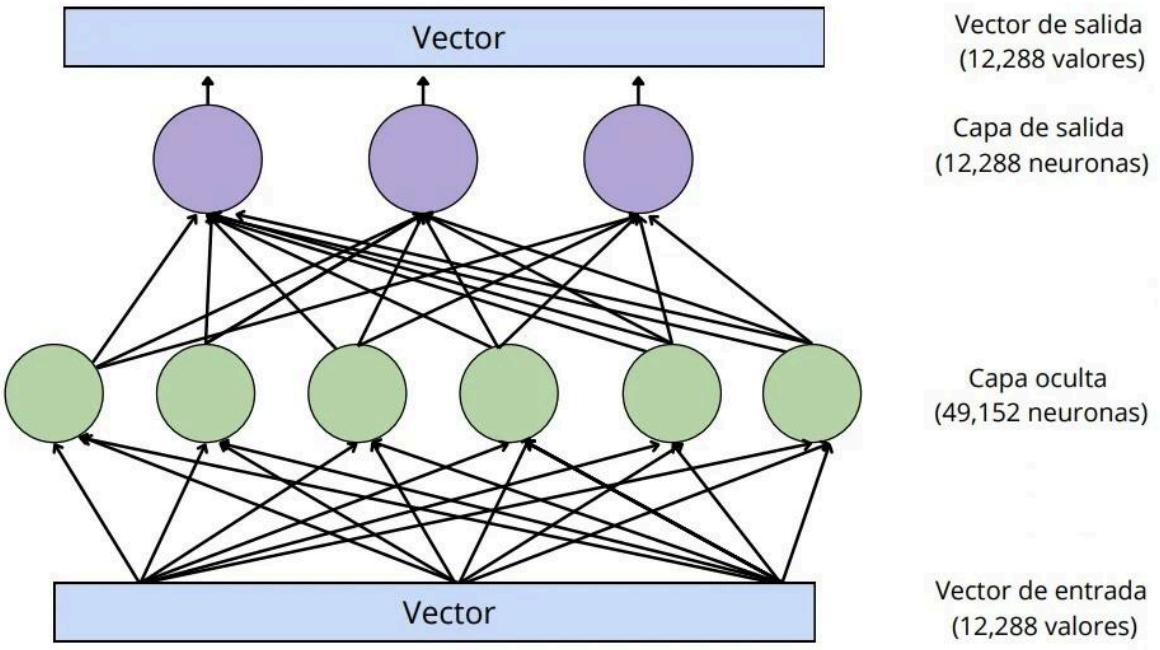
Por ejemplo, ¿cómo supo el modelo que la siguiente palabra debía ser el nombre de alguien y no otro tipo de palabra? Es sencillo pensar en oraciones similares en las que **Mary** no sería una buena predicción de siguiente palabra, como podría ser el caso de la oración "cuando Mary y John fueron al restaurante, John le dio sus llaves al", en la cual la siguiente palabra lógica sería "valet".

En teoría, con suficiente investigación, los científicos de la computación podrían revelar y explicar pasos adicionales en el proceso de razonamiento de GPT-2. Eventualmente, podrían desarrollar un entendimiento exhaustivo de cómo GPT-2 decidió que **Mary** es la siguiente palabra más probable para esa oración. Sin embargo, podría tomar meses o incluso años de esfuerzos adicionales únicamente para entender la predicción de una sola palabra.

Los modelos de lenguaje subyacentes en ChatGPT —GPT-3 y GPT-4— son significativamente más grandes y complejos que GPT-2. Aquellos son capaces de realizar tareas que requieren un razonamiento más complejo que la simple tarea de completar oraciones estudiada por el equipo de Redwood. Por tanto, explicar por completo cómo funcionan estos sistemas será un enorme proyecto que es poco probable que la humanidad complete en el futuro próximo.

El paso de propagación hacia adelante

Una vez que los cabezales de atención han transferido información entre los vectores de palabras, hay una red *feed-forward*[3] que "piensa" sobre cada vector e intenta predecir la siguiente palabra. En esta etapa no hay intercambio de información entre palabras: la capa *feed-forward* analiza cada palabra por separado. Sin embargo, la capa *feed-forward* sí tiene acceso a cualquier información que haya sido previamente copiada por un cabezal de atención. Esta es la estructura de la capa *feed-forward* en la versión más grande de GPT-3:



Los círculos verdes y morados son neuronas: funciones matemáticas que calculan una suma ponderada de sus inputs.[4]

Lo que hace poderosa a la capa *feed-forward* es su enorme número de conexiones. Hemos dibujado esta red con tres neuronas en la capa de salida y seis neuronas en la capa oculta; sin embargo, las capas *feed-forward* de GPT-3 son mucho más grandes: 12,288 neuronas en la capa de salida (que corresponden a los vectores de 12,288 dimensiones del modelo) y 49,152 neuronas en la capa oculta.

Entonces, en la versión más grande de GPT-3 hay 49,152 neuronas en la capa oculta con 12,288 inputs (y, por tanto, 12,288 parámetros) por cada neurona. Asimismo, hay 12,288 neuronas de salida con 49,152 valores de entrada (y, por tanto, 49,152 parámetros) por cada neurona. Esto significa que cada capa *feed-forward* tiene 49,152 * 12,288 + 12,288 * 49,152 = 1,200 millones de parámetros. Y hay 96 capas *feed-forward*, para un total de 1,200 millones * 96 = ¡116 mil millones de parámetros! Esto representa casi dos terceras partes del total general de 175 mil millones de parámetros de GPT-3.

En un *paper* de 2020, investigadores de la Universidad de Tel Aviv encontraron que las capas *feed-forward* funcionan mediante emparejamiento de patrones: cada neurona en la capa oculta se empareja con un patrón específico del texto de entrada. Aquí hay algunos de los patrones que fueron emparejados por neuronas en una versión de GPT-2 con 16 capas. Si bien es importante aclarar que los investigadores de la Universidad de Tel Aviv realizaron su estudio con palabras y oraciones emparejadas en inglés, a continuación te mostraremos y analizaremos algunos de los patrones que se detectaron, solo que traducidos al español:

- En la capa 1 , una neurona emparejó secuencias de palabras que terminaban en "sustitutos".
- En la capa 6, una neurona emparejó secuencias relacionadas con lo militar, o the military en inglés, con las terminaciones "base" o "bases", ya que, en inglés, base militar se dice, en estricto orden, military base.
- En la capa 13, una neurona emparejó secuencias terminadas en intervalos de tiempo, tales como "entre las 3 pm y las 7" o "de las 7:00 pm del viernes hasta".
- En la capa 16, una neurona emparejó secuencias relacionadas con programas televisivos, tales como "la versión diurna original de la NBC, archivada" o "la visualización en diferido añadió un 57 % al episodio".

Como puedes ver, los patrones se volvieron más abstractos en las capas posteriores. Mientras que las primeras tendieron a emparejar palabras específicas, las posteriores emparejaron frases comprendidas dentro de categorías semánticas más amplias, como programas televisivos o intervalos de tiempo.

Esto es interesante porque, como lo mencionamos previamente, la capa *feed-forward* examina únicamente una palabra a la vez. Por tanto, cuando clasifica la secuencia "la versión diurna original de la NBC, archivada" en relación con la televisión, solo tiene acceso al vector correspondiente a **archivada** y no a los vectores de palabras como **NBC** o **diurna**.

Se supone que la capa *feed-forward* puede darse cuenta de que **archivada** es parte de una secuencia relacionada con la televisión porque previamente los cabezales de atención movieron información contextual hacia el vector correspondiente a **archivada.**

Cuando una neurona empareja alguno de estos patrones, añade información al vector. Si bien dicha información no siempre es fácil de interpretar, en muchos casos funciona pensar en ella como una predicción tentativa sobre la siguiente palabra.

Las redes *feed-forward* razonan con matemática vectorial

Investigaciones recientes de la Universidad de Brown revelaron un elegante ejemplo de cómo las capas *feed-forward* ayudan a predecir la siguiente palabra. Anteriormente discutimos que las investigaciones desarrolladas en el proyecto word2vec de Google mostraron la posibilidad de usar aritmética vectorial para razonar por analogía. Por ejemplo, **Berlín** – **Alemania** + **Francia** = **París.**

Los investigadores de Brown encontraron que las capas *feed-forward* a veces utilizan este preciso método para predecir la siguiente palabra. Por ejemplo, examinaron cómo respondió GPT-2 al siguiente *prompt* en inglés: **"Q: What is the capital of France? A: Paris Q: What is the capital of Poland? A:"**. Un *prompt* equivalente en español podría ser: "Pregunta: ¿cuál es la capital de Francia? Respuesta: París; Pregunta: ¿cuál es la capital de Polonia? Respuesta**:".**

El equipo estudió una versión de GPT-2 con 24 capas. Después de cada una, los investigadores de Brown probaron el modelo para observar su mejor predicción en el siguiente token. Para el caso de las primeras 15 capas, la predicción principal era una palabra aparentemente aleatoria. Entre las capas 16 y 19, el modelo empezó a predecir que la siguiente palabra sería **Polonia,** lo cual no es correcto, pero iba

acercándose. Después, en la capa 20, la predicción principal cambió a **Varsovia** —la respuesta correcta—y se mantuvo así en las últimas cuatro capas.

Los investigadores de Brown encontraron que la capa *feed-forward* número 20 convirtió Polonia en Varsovia añadiendo un vector que mapea vectores de países con sus correspondientes capitales. Al añadir el mismo vector a **China**, este dio como resultado **Beijing**.

Las capas *feed-forward* en el mismo modelo utilizaron aritmética vectorial para transformar palabras en minúsculas en palabras con mayúsculas, así como palabras en tiempo presente en sus equivalentes en tiempo pasado.

Las capas de atención y las capas *feed-forward* tienen trabajos diferentes

Hasta este momento hemos visto dos ejemplos reales de predicciones de palabras hechas por GPT-2: los cabezales de atención que ayudan a predecir que **John** le dio una bebida a **Mary** y una capa *feed-forward* que ayuda a predecir que **Varsovia** es la capital de **Polonia.**

En el primer caso, **Mary** estaba incluida en el *prompt* dado por el usuario. Pero en el segundo caso, **Varsovia** no lo estaba. Más bien, GPT-2 tuvo que "recordar" el hecho de que **Varsovia** es la capital de **Polonia** —información que aprendió de los datos de entrenamiento, conocidos como **training data** en idioma inglés—.

Cuando los investigadores de Brown desactivaron la capa *feed-forward* que convirtió **Polonia** en **Varsovia,** el modelo ya no pudo predecir a **Varsovia** como la siguiente palabra. Pero, interesantemente, si al principio del *prompt* original añadían (en inglés) la oración "La capital de Polonia es Varsovia", entonces GPT-2 podía responder la pregunta nuevamente. Esto se debe probablemente a que GPT-2 utilizó cabezales de atención para copiar el nombre **Varsovia** del principio del *prompt*.

De manera general, la división de trabajo sucede así: los cabezales de atención recuperan información de palabras previas en el *prompt*, mientras que las capas *feed-forward* permiten que los modelos de lenguaje "recuerden" información que no está en el *prompt*.

En efecto, una forma de pensar en las capas *feed-forward* es como una base de datos de información que el modelo ha aprendido de sus datos de entrenamiento. Es más probable que las primeras capas *feed-forward* codifiquen hechos simples relacionados con palabras específicas, tales como que "Trump con frecuencia viene después de Donald". Las capas subsecuentes codifican relaciones más complejas, tales como "suma este vector para convertir un país en su capital".

Cómo se entrenan los modelos de lenguaje

Muchos de los primeros algoritmos de *machine learning* requerían que los ejemplos utilizados para su entrenamiento fueran etiquetados a mano por humanos. Por ejemplo, los datos de entrenamiento podrían haber sido fotos de perros o gatos con la etiqueta "perro" o "gato" proveída por un humano respecto de cada foto. La necesidad de humanos para etiquetar datos hacía difícil y caro crear conjuntos de datos lo suficientemente grandes para entrenar modelos potentes.

Una innovación clave de los LLM consiste en que no necesitan datos explícitamente etiquetados; en cambio, aprenden a través de intentar predecir la siguiente palabra en un texto ordinario. Prácticamente cualquier material escrito —desde páginas de Wikipedia hasta artículos de noticias o código de programación— es adecuado para entrenar estos modelos.

Por ejemplo, a un LLM se le podría dar el input "Me gusta el café con crema y", y se supondría que debería predecir "azúcar" como la siguiente palabra. Un modelo de lenguaje recién inicializado sería muy malo en esto porque cada uno de sus parámetros (o pesos, del inglés, weights) —175 mil millones de ellos en la versión más potente de GPT-3—empezaría básicamente como un número aleatorio.

Sin embargo, conforme el modelo va aprendiendo de más y más ejemplos —cientos de miles de millones de palabras—, esos parámetros se ajustan gradualmente para hacer predicciones cada vez mejores.

A continuación, incluimos una analogía para ilustrar mejor este funcionamiento. Supongamos que vas a tomar una ducha y quieres que la temperatura sea perfecta: no demasiado caliente y no demasiado fría. Nunca has utilizado antes esta regadera, así

que giras la perilla hacia una dirección aleatoria y sientes la temperatura del agua. Si está demasiado caliente, giras la perilla hacia el otro lado y si está demasiado fría, la giras en la dirección contraria. Conforme te acercas a la temperatura deseada, los ajustes que haces a la perilla se van haciendo más pequeños.

Ahora hagamos un par de cambios a la analogía. Primero, imagina que la regadera tiene 50,257 grifos en vez de solo uno. Cada grifo corresponde a una palabra diferente, como **el**, **gato** o **banco**. Tu objetivo es que el agua salga únicamente del grifo correspondiente a la siguiente palabra en una frase.

En segundo lugar, supón que hay un laberinto de tuberías interconectadas *detrás* de los grifos y que a su vez dichas tuberías tienen un montón de válvulas. Por tanto, si el agua sale a través del grifo equivocado, no simplemente ajustas la perilla del grifo de la regadera, sino que despliegas a un ejército de ardillas inteligentes que recorren cada tubería hacia atrás y ajustan cada válvula que encuentran en el camino.

Lo anterior se vuelve complicado porque, muchas veces, la misma tubería alimenta varios grifos. Por tanto, se necesita un análisis cuidadoso para descubrir qué válvulas se deben apretar, cuáles se deben aflojar y en qué medida.

Obviamente, este ejemplo rápidamente se vuelve absurdo si lo tomas demasiado literal. No sería realista o útil construir una red de tuberías con 175 mil millones de válvulas; pero, gracias a la Ley de Moore, las computadoras pueden operar a este nivel de escala (y de hecho lo hacen).

Todas las partes de los LLM que hemos discutido en este artículo hasta este punto —las neuronas en las capas *feed-forward y* los cabezales de atención que mueven información contextual entre palabras— son implementadas como una cadena de simples funciones matemáticas (en su mayoría multiplicaciones de matrices) cuyo comportamiento es determinado por parámetros (pesos) ajustables. Así como las ardillas en el ejemplo aflojan y aprietan las válvulas para controlar el flujo de agua, el algoritmo de entrenamiento incrementa o reduce los parámetros del modelo de lenguaje para controlar cómo fluye la información a través de la red neuronal.

El proceso de entrenamiento consta de dos pasos. Primero, hay un "paso hacia adelante", en el que se abre el agua y se revisa si sale por el grifo correcto. Luego cierras el agua y hay un "paso hacia atrás", en el que las ardillas recorren cada tubería apretando y aflojando válvulas. En redes neuronales digitales, el rol de las ardillas es llevado a cabo por un algoritmo llamado **retropropagación**, o **backpropagation** en inglés, que consiste en "recorrer hacia atrás" la red, utilizando un cálculo para estimar qué tanto cambiar cada peso.[5]

Completar este proceso —hacer un paso hacia adelante con un ejemplo y después un paso hacia atrás para mejorar el desempeño de la red en ese ejemplo— requiere cientos de miles de millones de operaciones matemáticas. Además, entrenar a un modelo tan grande como GPT-3 requiere repetir el proceso miles de millones de veces, una vez por cada palabra de los datos de entrenamiento.[6] OpenAI estima que requirió más de 300 mil millones de billones de cálculos de coma flotante para entrenar a GPT-3. Eso significa meses de trabajo para docenas de chips de computadora de alta gama.

El sorprendente desempeño de GPT-3

Tal vez te sorprenda que el proceso de entrenamiento funcione tan bien como funciona. ChatGPT puede realizar todo tipo de tareas complejas: redactar ensayos, formular analogías e incluso escribir código computacional. Entonces, ¿cómo es que un mecanismo de aprendizaje tan simple produce un modelo tan poderoso?

Una de las razones es la *escalabilidad.* No se puede dejar de enfatizar en la enorme cantidad de ejemplos que ve un modelo como GPT-3, que fue entrenado con un corpus de aproximadamente 500 mil millones de palabras. En comparación, un niño humano promedio se habrá encontrado con alrededor de 100 millones de palabras a la edad de diez años.

Durante los últimos 5 años, OpenAI ha incrementado de manera constante el tamaño de sus modelos de lenguaje. En un *paper* de 2020 ampliamente leído, OpenAI reportó que la precisión de sus modelos de lenguaje escaló exponencialmente con el tamaño del modelo, el tamaño del conjunto de datos y la capacidad de procesamiento usada para el entrenamiento, con algunas tendencias que abarcan más de siete potencias.

Cuanto más grandes se volvían los modelos, mejor se desempeñaban en tareas relacionadas con el lenguaje. Sin embargo, esto solo era cierto si incrementaban la cantidad de datos de entrenamiento por un factor similar. Para entrenar modelos más grandes con más datos, necesitas mucha más capacidad de procesamiento.

El primer LLM de OpenAI, GPT-1, fue lanzado en 2018. GPT-1 utilizó vectores de 768 dimensiones y tenía 12 capas para un total de 117 millones de parámetros. Unos meses después, OpenAI lanzó GPT-2. La versión más grande de GPT-2 tenía vectores de 1,600 dimensiones, 48 capas y un total de 1,500 millones de parámetros.

En 2020, OpenAI lanzó GPT-3, el cual incluyó vectores de 12,288 dimensiones y 96 capas para un total de 175 mil millones de parámetros.

Finalmente, el año pasado, OpenAI lanzó GPT-4. La compañía no ha publicado ningún detalle sobre la arquitectura de GPT-4; sin embargo, existe un amplio consenso de que es significativamente más grande que GPT-3.

Cada modelo no solo aprendió más información que sus predecesores más pequeños, sino que también se desempeñó mejor en tareas que requirieron algún tipo de razonamiento abstracto.

Por ejemplo, considera la siguiente historia:

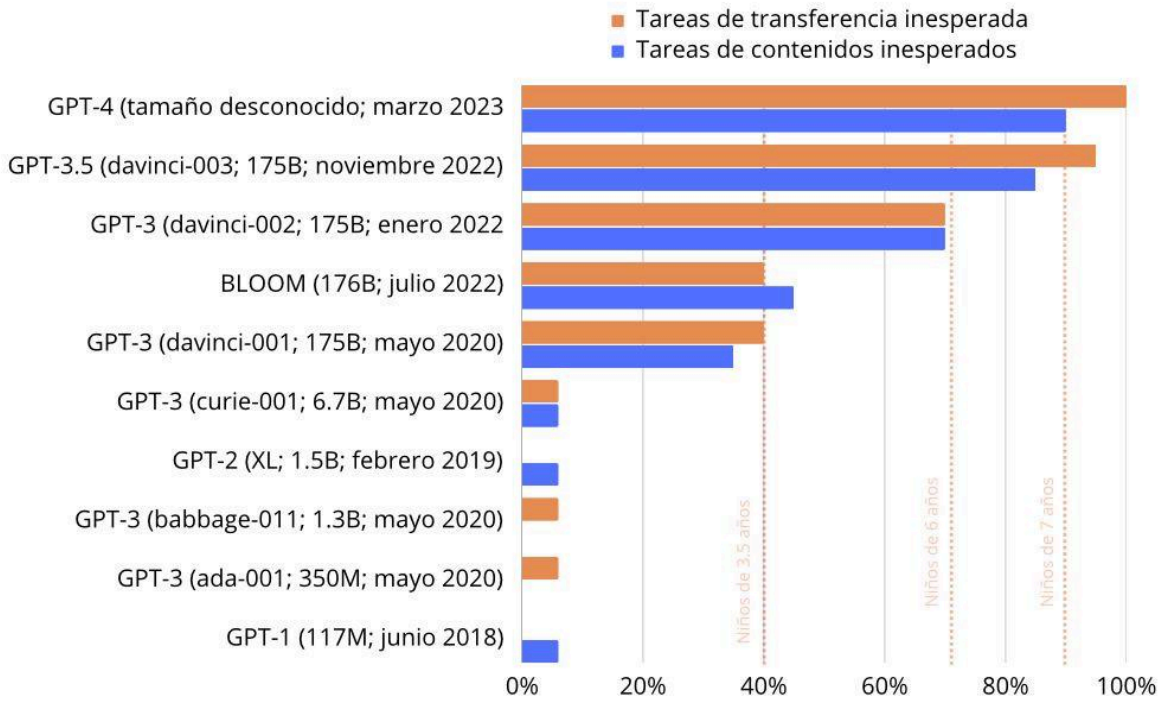
> Aquí hay una bolsa llena de palomitas. No hay chocolate dentro de la bolsa. Sin embargo, la etiqueta de la bolsa dice "chocolate", en vez de "palomitas". Sam encuentra la bolsa. Ella nunca ha visto la bolsa y no puede ver qué hay en su interior. Sam lee la etiqueta.

Probablemente puedes adivinar que Sam cree que la bolsa contiene chocolate y que se sorprenderá al descubrir palomitas en su interior. Los psicólogos le llaman "teoría de la mente" a esta capacidad de razonar sobre los estados mentales de otras personas. La mayoría de las personas adquieren esta capacidad desde el momento en que están en la escuela primaria. No hay consenso entre expertos sobre si otros

animales no humanos (como los chimpancés) tienen teoría de la mente, pero sí existe un consenso general de que es importante para la cognición social humana.

A principios de 2023, el psicólogo de Stanford, Michal Kosinski, publicó una investigación que examinaba la habilidad de los LLM para resolver tareas relacionadas con teoría de la mente. Para ello, proporcionó extractos de texto (como el que citamos arriba) a varios modelos de lenguaje y, posteriormente, les pidió completar oraciones tales como "ella cree que la bolsa está llena de". La respuesta correcta a lo anterior es "chocolate"; sin embargo, un modelo de lenguaje poco sofisticado podría responder "palomitas" o algo diferente.

GPT-1 y GPT-2 reprobaron la prueba de Kosinski, pero la primera versión de GPT-3, lanzada en el 2020, acertó en casi el 40 por ciento de las veces, un nivel de desempeño que Kosinski compara con el de un niño de tres años. La versión más reciente de GPT-3, lanzada en noviembre de 2022, mejoró su desempeño hasta alcanzar un 90 %, poniéndose a la par de un niño de siete años. GPT-4 respondió correctamente un 95 % de las preguntas de teoría de la mente.



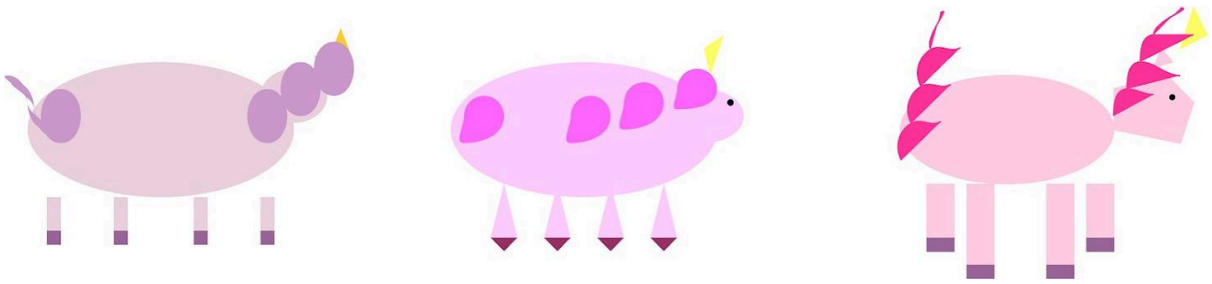
“Dado que no existe ningún indicio de que la habilidad que se asemeja a una teoría de la mente haya sido deliberadamente programada en estos modelos, ni investigación que demuestre que los científicos saben cómo alcanzar eso, es probable que dicha habilidad que se aproxima a una teoría de la mente haya surgido espontánea y autónomamente como consecuencia de la creciente habilidad de lenguaje de los modelos”, escribió Kosinski (la traducción es nuestra).

Es importante aclarar que no todos los investigadores coinciden en que estos resultados constituyen evidencia de teoría de la mente: por ejemplo, pequeños cambios a la tarea de la creencia falsa llevaron a un desempeño mucho peor por parte de GPT-3; asimismo, GPT-3 presenta un desempeño más variable ante otras tareas relacionadas con la medición de teoría de la mente. Como uno de nosotros (Sean) ha escrito, podría ser que el desempeño exitoso es atribuible a variables de confusión en la tarea (una suerte de efecto “Clever Hans”, solo que en modelos de lenguaje en lugar de caballos).

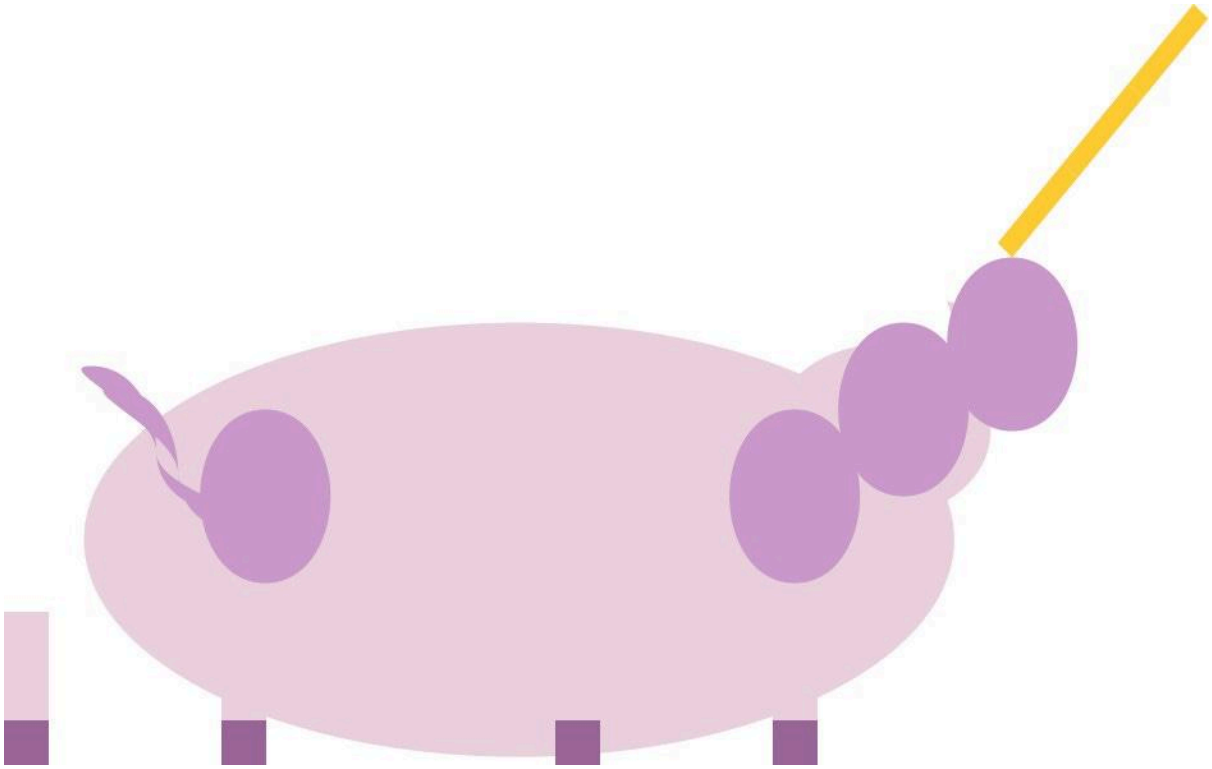
No obstante lo anterior, el desempeño casi humano de GPT-3 en varias tareas diseñadas para medir la teoría de la mente hubiera sido impensable tan solo unos años atrás. Y es consistente con la idea de que modelos más grandes son generalmente mejores resolviendo tareas que requieren razonamiento de alto nivel.

Este es solo uno de muchos ejemplos de modelos de lenguaje que aparentemente han desarrollado capacidades de razonamiento de alto nivel de manera espontánea. En abril de 2023, investigadores de Microsoft publicaron un *paper* en el que argumentan que GPT-4 mostró las primeras pistas prometedoras de inteligencia artificial general: la habilidad de pensar en una forma sofisticada, similar a la de los humanos.

Por ejemplo, un investigador le pidió a GPT-4 que dibujara un unicornio utilizando TiKZ, un lenguaje de programación de gráficos poco conocido. GPT-4 respondió con unas cuantas líneas de código, las cuales el investigador posteriormente ingresó en el programa TiKZ. Las imágenes resultantes eran rudimentarias, pero mostraban señales claras de que GPT-4 tenía cierto entendimiento de cómo lucen los unicornios.



Los investigadores pensaron que GPT-4 tal vez había memorizado de alguna manera el código necesario para dibujar un unicornio a partir de sus datos de entrenamiento, por lo que le dieron un nuevo desafío: alteraron el código del unicornio para remover el cuerno y mover otras partes de su cuerpo. Posteriormente, le pidieron a GPT-4 que pusiera de nuevo el cuerno. GPT-4 respondió colocando el cuerno en el lugar correcto:



GPT-4 fue capaz de hacer esto aun cuando los datos de entrenamiento de la versión probada por los investigadores estaban compuestos únicamente por texto, es decir, no había imágenes en sus datos de entrenamiento. Al parecer, GPT-4 aprendió a

razonar sobre la forma del cuerpo de un unicornio después de su entrenamiento, a partir de una enorme cantidad de texto.

Por ahora no tenemos una visión real de cómo es que los LLM son capaces de realizar tareas como esta. Algunas personas argumentan que ejemplos como el anterior demuestran que los modelos están empezando a verdaderamente entender el significado de las palabras contenidas en sus datos de entrenamiento. Otros insisten en que los modelos de lenguaje son "loros estocásticos" que simplemente repiten secuencias de palabras cada vez más complejas, sin realmente entenderlas.

Este debate apunta hacia una profunda tensión filosófica que probablemente sea imposible de resolver. Sin embargo, nosotros pensamos que es importante enfocarnos en el desempeño *empírico* de modelos como GPT-3. Si un modelo de lenguaje es capaz de consistentemente responder de manera correcta a un tipo particular de preguntas y si los investigadores confían en que han controlado variables de confusión (por ejemplo, asegurándose de que el modelo de lenguaje no haya sido expuesto a dichas preguntas durante su entrenamiento), entonces ese es un resultado interesante e importante, ya sea que entienda el lenguaje, o no, exactamente en el mismo sentido en que lo entienden las personas.

Otra posible razón por la cual el entrenamiento para predecir el siguiente token funciona tan bien es porque el lenguaje en sí mismo es predecible. Con frecuencia (aunque no siempre), los patrones en el lenguaje están conectados con patrones en el mundo físico. Por tanto, cuando un modelo de lenguaje aprende sobre relaciones entre palabras, a menudo está también implícitamente aprendiendo sobre relaciones en el mundo.

Además, las predicciones pueden ser fundamentales tanto para la inteligencia biológica como para la inteligencia artificial. En la visión de filósofos como Andy Clark, el cerebro humano puede entenderse como una "máquina de predicción", cuyo trabajo principal es hacer predicciones sobre nuestro ambiente, predicciones que después puedan ser usadas para navegar exitosamente dicho ambiente. Intuitivamente, hacer buenas predicciones se beneficia de buenas *representaciones*; es decir, es más probable que navegues de manera exitosa teniendo un mapa preciso que con un mapa impreciso. El mundo es grande y complejo, por lo que hacer

predicciones ayuda a los organismos a orientarse y a adaptarse eficientemente a dicha complejidad.

Tradicionalmente, un reto importante para construir modelos de lenguaje era encontrar la manera más útil de representar diferentes palabras, especialmente porque los significados de las palabras dependen mucho del contexto. El enfoque de predicción de la siguiente palabra permite a los investigadores eludir este espinoso rompecabezas teórico, convirtiéndolo en un problema empírico. Resulta que, si les proveemos de suficiente información y capacidad de procesamiento, los modelos de lenguaje consiguen aprender mucho sobre cómo funciona el lenguaje humano simplemente buscando cómo predecir mejor la siguiente palabra. La desventaja de esto es que terminamos con sistemas cuyo funcionamiento interno no entendemos del todo.

Tim Lee formó parte del staff de Ars del 2017 al 2021. Recientemente lanzó una nueva newsletter, Understanding AI, en la cual explora cómo funciona la IA y cómo está cambiando nuestro mundo. Aquí puedes suscribirte.

Sean Trott es un Asistente de Profesor en la Universidad de California en San Diego, donde conduce investigaciones sobre entendimiento del lenguaje en humanos y grandes modelos de lenguaje. Escribe sobre estos y otros temas en su newsletter The Counterfactual.

Rubén Álvarez Escobar es abogado por la UDLAP, interesado en descubrir formas innovadoras y responsables de usar la Inteligencia Artificial en su práctica como asesor legal de empresas y emprendimientos tecnológicos en México.

Sylvia Elena Rodríguez es traductora, editora y activista por los derechos lingüísticos y la democratización del conocimiento.

Elen Irazabal es doctoranda por la UAM y el CSIC en Derecho, Gobierno y Políticas Públicas. Autora del libro "La Inteligencia Artificial explicada para abogados" y forma a abogados en Inteligencia Artificial. Profesora de ética de la Inteligencia Artificial de la Carlos III en el Máster de Computación.

Enrique Onieva es Doctor en Informática y actualmente es profesor e investigador en Inteligencia Artificial en la Universidad de Deusto (Bilbao-España).

1

Técnicamente, los LLM operan con fragmentos de palabras llamados tokens, pero ignoraremos este detalle de implementación para mantener el tamaño del artículo dentro de lo manejable.

2

Técnicamente, la versión original de ChatGPT tiene como base a GPT-3.5, un sucesor de GPT-3 que fue sometido a un proceso llamado Aprendizaje por Refuerzo con Retroalimentación Humana (RLHF, conforme a las siglas del término original en inglés). OpenAI no ha publicado todos los detalles arquitectónicos de este modelo, por lo que en este artículo nos centraremos en GPT-3, la última versión que OpenAI ha descrito en detalle.

3

La red *feed-forward* es también conocida como un perceptrón multicapa. Científicos informáticos han estado experimentando con este tipo de redes neuronales desde los años 60.

4

Técnicamente, después de que una neurona calcula una suma ponderada de sus inputs, pasa el resultado a una función de activación. Vamos a ignorar este detalle de implementación, pero si quieres una explicación completa de cómo funcionan las neuronas, puedes leer este artículo de Tim de 2018 en el que lo explica.

5

Si deseas aprender más sobre retropropagación, échale un vistazo a este artículo de Tim de 2018 en el que explica cómo funcionan las redes neuronales.

6

En la práctica, con fines de eficiencia computacional, el entrenamiento es con frecuencia hecho en lotes, de manera que el *software* puede hacer el pase hacia adelante sobre 32 mil tokens antes de hacer un pase hacia atrás.
