

Modelos matemáticos del lenguaje natural (p.2 post-2022)

Mauricio Velasco*
Centro de Matemáticas (CMAT)
Facultad de Ciencias
Universidad de la Republica

Julio 2025

Modelos de lenguaje actuales (2025)

En 2022 ChatGPT mostró que los patrones estadísticos del lenguaje natural pueden ser capturados sorprendentemente bien por algunos modelos.

Modelos de lenguaje actuales (2025)

En 2022 ChatGPT mostró que los patrones estadísticos del lenguaje natural pueden ser capturados sorprendentemente bien por algunos modelos.

*La idea clave detras de esta familia de modelos es la **función de atención**, que ha revolucionado la manera en la que se procesa informacion.*

Modelos de lenguaje actuales (2025)

En 2022 ChatGPT mostró que los patrones estadísticos del lenguaje natural pueden ser capturados sorprendentemente bien por algunos modelos.

*La idea clave detras de esta familia de modelos es la **función de atención**, que ha revolucionado la manera en la que se procesa informacion.*

Concretamente, en esta segunda charla:

- 1 Describiremos la función de atención y su uso en la construcción de modelos de lenguaje natural.
- 2 Estudiaremos una implementación concreta de un GPT muy simple (SLM).
- 3 Discutiremos usos y capacidades de arquitecturas basadas en atención: traducción de textos, búsquedas contextuadas en bases de datos e instruction following.

Retomemos el modelo:

Fijamos un alfabeto \mathcal{W} que representa nuestro diccionario de palabras.

Retomemos el modelo:

Fijamos un alfabeto \mathcal{W} que representa nuestro diccionario de palabras. Queremos aprender

$$p(x_m | x_{[m-\ell, m-1]})$$

Retomemos el modelo:

Fijamos un alfabeto W que representa nuestro diccionario de palabras. Queremos aprender

$$p(x_m | x_{[m-\ell, m-1]})$$

una familia de **distribuciones de probabilidad sobre W** que dependa del contexto $x_{[m-\ell, m-1]} := (x_{m-\ell}, \dots, x_{m-1})$.

Retomemos el modelo:

Fijamos un alfabeto W que representa nuestro diccionario de palabras. Queremos aprender

$$p(x_m | x_{[m-\ell, m-1]})$$

una familia de **distribuciones de probabilidad sobre W** que dependa del contexto $x_{[m-\ell, m-1]} := (x_{m-\ell}, \dots, x_{m-1})$.

Asumiremos:

$$p(x_m = w_j | x_{[m-\ell, m-1]}) = \mathbb{B} \left(F_\theta(x_{[m-\ell, m-1]}) \right)_j$$

donde $F_\theta : W^\ell \rightarrow \mathbb{R}^{|W|}$ es una función que depende de parámetros θ y \mathbb{B} es la función de Boltzmann.

Limitación de word2vec

Pregunta.

Por qué no simplemente entrenar un word2vec?

Limitación de word2vec

Pregunta.

Por qué no simplemente entrenar un word2vec?

La geometría definida por un word2vec tiene una limitación fundamental: A cada palabra le asigna un vector estático. Esto ignora el hecho de que **la misma palabra puede tener significados distintos en contextos distintos.**

Limitación de word2vec

Pregunta.

Por qué no simplemente entrenar un word2vec?

La geometría definida por un word2vec tiene una limitación fundamental: A cada palabra le asigna un vector estático. Esto ignora el hecho de que **la misma palabra puede tener significados distintos en contextos distintos.**

Ejemplo:

- *Mi **gato** duerme todo el día.*
- *Necesitamos un **gato** para cambiar la rueda del auto.*

Limitación de word2vec

Pregunta.

Por qué no simplemente entrenar un word2vec?

La geometría definida por un word2vec tiene una limitación fundamental: A cada palabra le asigna un vector estático. Esto ignora el hecho de que **la misma palabra puede tener significados distintos en contextos distintos.**

Ejemplo:

- *Mi **gato** duerme todo el día.*
- *Necesitamos un **gato** para cambiar la rueda del auto.*

Para poder capturar los patrones estadísticos del lenguaje tenemos que permitir esa multiplicidad de significados.

Limitación de word2vec

Pregunta.

Por qué no simplemente entrenar un word2vec?

La geometría definida por un word2vec tiene una limitación fundamental: A cada palabra le asigna un vector estático. Esto ignora el hecho de que **la misma palabra puede tener significados distintos en contextos distintos.**

Ejemplo:

- *Mi **gato** duerme todo el día.*
- *Necesitamos un **gato** para cambiar la rueda del auto.*

Para poder capturar los patrones estadísticos del lenguaje tenemos que permitir esa multiplicidad de significados. Lo ideal sería que, dado un contexto X cada palabra $w \in W$ tuviera un vector distinto que la representa.

P A R T E I:

La función de atención (self-attention).

Word embeddings contextualizados

La manera más sencilla de construir word embeddings contextualizados es modificando cada vector mediante una suma ponderada de los vectores de su contexto.

Word embeddings contextualizados

La manera más sencilla de construir word embeddings contextualizados es modificando cada vector mediante una suma ponderada de los vectores de su contexto.

Los coeficientes grandes de esta ponderación indican a qué otras palabras es necesario poner atención para poder construir una representación contextualizada.

Word embeddings contextualizados

La manera más sencilla de construir word embeddings contextualizados es modificando cada vector mediante una suma ponderada de los vectores de su contexto.

Los coeficientes grandes de esta ponderación indican a qué otras palabras es necesario poner atención para poder construir una representación contextualizada.

Esta idea nos lleva naturalmente al concepto de **atención**, que describiremos a continuación.

Bases de datos vectorizadas

Para describir atención usaremos terminología de base de datos.

Bases de datos vectorizadas

Para describir atención usaremos terminología de base de datos.

Una **base de datos** es una abstracción de un diccionario (una implementación es la clase dict de python o el formato json). En un diccionario buscamos un término (**key**) y encontramos en el su definición (**value**) (1950s-1970s).

Una base de datos es $\{(k_1, v_1), \dots, (k_N, v_N)\}$

Bases de datos vectorizadas

Para describir atención usaremos terminología de base de datos.

Una **base de datos** es una abstracción de un diccionario (una implementación es la clase dict de python o el formato json). En un diccionario buscamos un término (**key**) y encontramos en el su definición (**value**) (1950s-1970s).

Una base de datos es $\{(k_1, v_1), \dots, (k_N, v_N)\}$

Posteriormente en bases de datos relacionales se introduce el concepto de preguntas (**queries**) que retornan conjuntos de valores mediante filtrado de los **keys** (SQL 1970s)

Bases de datos vectorizadas

Para describir atención usaremos terminología de base de datos.

Una **base de datos** es una abstracción de un diccionario (una implementación es la clase dict de python o el formato json). En un diccionario buscamos un término (**key**) y encontramos en el su definición (**value**) (1950s-1970s).

Una base de datos es $\{(k_1, v_1), \dots, (k_N, v_N)\}$

Posteriormente en bases de datos relacionales se introduce el concepto de preguntas (**queries**) que retornan conjuntos de valores mediante filtrado de los **keys** (SQL 1970s)

*En nuestro caso usaremos el paradigma de **bases de datos vectorizadas** en las que queries, keys y values son vectores.*

Bases de datos vectorizadas

Para describir atención usaremos terminología de base de datos.

Una **base de datos** es una abstracción de un diccionario (una implementación es la clase dict de python o el formato json). En un diccionario buscamos un término (**key**) y encontramos en el su definición (**value**) (1950s-1970s).

Una base de datos es $\{(k_1, v_1), \dots, (k_N, v_N)\}$

Posteriormente en bases de datos relacionales se introduce el concepto de preguntas (**queries**) que retornan conjuntos de valores mediante filtrado de los **keys** (SQL 1970s)

*En nuestro caso usaremos el paradigma de **bases de datos vectorizadas** en las que queries, keys y values son vectores. Típicamente el $\vec{q}, \vec{k}_j \in \mathbb{R}^d$ y $\vec{v}_j \in \mathbb{R}^v$.*

Bases de datos vectorizadas

*En nuestro caso usaremos el paradigma de **bases de datos vectorizadas** en las que queries, keys y values son vectores.*

Bases de datos vectorizadas

*En nuestro caso usaremos el paradigma de **bases de datos vectorizadas** en las que queries, keys y values son vectores. Típicamente el $\vec{q}, \vec{k}_j \in \mathbb{R}^d$ y $\vec{v}_j \in \mathbb{R}^v$.*

La vectorización hace aparecer algunas estructuras adicionales nuevas. Dado un query \vec{q} podemos calcular el ángulo entre \vec{q} y cada uno de los \vec{k}_j mediante productos punto usando

$$\vec{q} \cdot \vec{k}_j = \|\vec{q}\| \|\vec{k}_j\| \cos(\theta_{q,k_j})$$

Querríamos construir un mecanismo de búsqueda "continuo" que cumpla:

- 1 Si \vec{q} es muy cercano a alguno de los k_i entonces el resultado del query sea muy cercano a v_i .
- 2 el resultado de cualquier query debe ser una combinación de los v_i que refleje la importancia relativa de los k_i en \vec{q} .

Querríamos construir un mecanismo de búsqueda "continuo" que cumpla:

- 1 *Si \vec{q} es muy cercano a alguno de los k_i entonces el resultado del query sea muy cercano a v_i .*
- 2 *el resultado de cualquier query debe ser una combinación de los v_i que refleje la importancia relativa de los k_i en \vec{q} .*

Definición.

El valor que resulta de auto-atención del query \vec{q} sobre la base de datos $\{\vec{k}_j, \vec{v}_j\}$ es

$$A(\vec{q}) := \sum_{j=1}^N \left(\frac{e^{\vec{q} \cdot \vec{k}_j}}{\sum_{i=1}^N e^{\vec{q} \cdot \vec{k}_i}} \right) \vec{v}_j$$

Definición.

El valor que resulta de auto-atención del query \vec{q} sobre la base de datos $\{\vec{k}_j, \vec{v}_j\}$ es

$$A(\vec{q}) := \sum_{j=1}^N \left(\frac{e^{\vec{q} \cdot \vec{k}_j}}{\sum_{i=1}^N e^{\vec{q} \cdot \vec{k}_i}} \right) \vec{v}_j$$

Notar que los coeficientes son un conjunto de N números positivos que suman uno, es decir una distribución de probabilidad.

Definición.

El valor que resulta de auto-atención del query \vec{q} sobre la base de datos $\{\vec{k}_j, \vec{v}_j\}$ es

$$A(\vec{q}) := \sum_{j=1}^N \left(\frac{e^{\vec{q} \cdot \vec{k}_j}}{\sum_{i=1}^N e^{\vec{q} \cdot \vec{k}_i}} \right) \vec{v}_j$$

Notar que los coeficientes son un conjunto de N números positivos que suman uno, es decir una distribución de probabilidad. Las magnitudes de esos coeficientes nos dicen que términos \vec{k}_j son importantes para determinar el resultado.

Definición.

El valor que resulta de auto-atención del query \vec{q} sobre la base de datos $\{\vec{k}_j, \vec{v}_j\}$ es

$$A(\vec{q}) := \sum_{j=1}^N \left(\frac{e^{\vec{q} \cdot \vec{k}_j}}{\sum_{i=1}^N e^{\vec{q} \cdot \vec{k}_i}} \right) \vec{v}_j$$

Notar que los coeficientes son un conjunto de N números positivos que suman uno, es decir una distribución de probabilidad. Las magnitudes de esos coeficientes nos dicen que términos \vec{k}_j son importantes para determinar el resultado. Esa es la justificación del término **atención** (a cuáles keys debemos atender para responder el query).

Word embeddings contextualizados

La manera más sencilla de construir word embeddings contextualizados es modificando cada vector mediante una suma ponderada de los vectores de su contexto.

Word embeddings contextualizados

La manera más sencilla de construir word embeddings contextualizados es modificando cada vector mediante una suma ponderada de los vectores de su contexto.

Los coeficientes grandes de esta ponderación indican a qué otras palabras es necesario poner atención para poder construir una representación contextualizada.

Word embeddings contextualizados

La manera más sencilla de construir word embeddings contextualizados es modificando cada vector mediante una suma ponderada de los vectores de su contexto.

Los coeficientes grandes de esta ponderación indican a qué otras palabras es necesario poner atención para poder construir una representación contextualizada.

Nuestros modelos de lenguaje deberán **aprender** una base de datos vectorizada con la propiedad de que el vector "contextualizado" se obtenga como una suma ponderada de los vectores del contexto mediante atención. La diferencia principal radica en que los mapas que deciden queries, keys y values deben ser aprendidos.

Concretamente, suponga que tenemos un embedding del contexto $X \in \mathbb{R}^{k \times \ell}$

$$X = \begin{pmatrix} | & | & \cdots & | \\ x_1 & x_2 & \cdots & x_\ell \\ | & | & \cdots & | \end{pmatrix}$$

y un nuevo vector $w \in \mathbb{R}^k$

Definición.

Para tres matrices de parámetros $\theta = (K, Q, V)$ y la **función de atención (self-attention)** dada por

$$A_\theta(X, w) := \sum_{j=1}^{\ell} \mathbb{B}((Qw) \cdot (Kx_j))_j Vx_j$$

con $Q, K \in \mathbb{R}^{b \times k}$ y $V \in \mathbb{R}^{h \times k}$.

Concretamente, suponga que tenemos un embedding del contexto $X \in \mathbb{R}^{k \times \ell}$

$$X = \begin{pmatrix} | & | & \cdots & | \\ x_1 & x_2 & \cdots & x_\ell \\ | & | & \cdots & | \end{pmatrix}$$

y un nuevo vector $w \in \mathbb{R}^k$

Definición.

Para tres matrices de parámetros $\theta = (K, Q, V)$ y la **función de atención (self-attention)** dada por

$$A_\theta(X, w) := \sum_{j=1}^{\ell} \mathbb{B}((Qw) \cdot (Kx_j))_j Vx_j$$

con $Q, K \in \mathbb{R}^{b \times k}$ y $V \in \mathbb{R}^{h \times k}$.

Se sigue que $A_\theta(X, w) : \mathbb{R}^{k \times \ell} \times \mathbb{R}^k \rightarrow \mathbb{R}^h$

Auto-Atención

Dado un contexto $X \in \mathbb{R}^{k \times \ell}$ y una palabra $w \in \mathbb{R}^k$ definimos tres matrices de parámetros $\theta = (K, Q, V)$ y la **función de atención (self-attention)** dada por

$$A_{\theta}(X, w) := \sum_{j=1}^{\ell} \mathbb{B}(\langle Qw, Kx_j \rangle)_j Vx_j$$

con $Q, K \in \mathbb{R}^{b \times k}$ y $V \in \mathbb{R}^{h \times k}$.

Dado un contexto $X \in \mathbb{R}^{k \times \ell}$ y una palabra $w \in \mathbb{R}^k$ definimos tres matrices de parámetros $\theta = (K, Q, V)$ y la **función de atención (self-attention)** dada por

$$A_{\theta}(X, w) := \sum_{j=1}^{\ell} \mathbb{B}(\langle Qw, Kx_j \rangle)_j Vx_j$$

con $Q, K \in \mathbb{R}^{b \times k}$ y $V \in \mathbb{R}^{h \times k}$.

Se sigue que

$$A_{\theta}(X, w) : \mathbb{R}^{k \times \ell} \times \mathbb{R}^k \rightarrow \mathbb{R}^h$$

Auto-atención

Imagine que hemos terminado el entrenamiento y encontrado las matrices $\theta^* := (Q^*, K^*, V^*)$.

$$A_{\theta}(X, w) := \sum_{j=1}^{\ell} \left(\frac{e^{(Q^*w) \cdot (K^*x_j)}}{\sum_i e^{(Q^*w) \cdot (K^*x_j)}} \right) V^*x_j$$

*Los productos internos $\langle Qw, Kx_j \rangle$ nos dan **coeficientes de participación** de cada x_j del contexto de w en la determinación del **vector contextualizado** que representa a w .*

Imagine que hemos terminado el entrenamiento y encontrado las matrices $\theta^* := (Q^*, K^*, V^*)$.

$$A_{\theta}(X, w) := \sum_{j=1}^{\ell} \left(\frac{e^{(Q^*w) \cdot (K^*x_j)}}{\sum_i e^{(Q^*w) \cdot (K^*x_i)}} \right) V^*x_j$$

*Los productos internos $\langle Qw, Kx_j \rangle$ nos dan **coeficientes de participación** de cada x_j del contexto de w en la determinación del **vector contextualizado** que representa a w .*

El origen de la notación K y V son **keys** y **values** en bases de datos (como un diccionario de python pero vectorial). El vector Qw representa la búsqueda (**query**) de w en la base de datos.
[2017 Vaswani "Attention is all you need"].

Auto-atención

Veamos los coeficientes de atención de **gato** en la siguiente frase

Ejemplo:

El gato está en el auto porque se rompió el neumático.

<i>El</i>	<i>0.08</i>
<i>gato</i>	<i>0.35</i>
<i>está</i>	<i>0.15</i>
<i>en</i>	<i>0.05</i>
<i>auto</i>	<i>0.25</i>
<i>neumatico</i>	<i>0.2</i>

Auto-atención

Veamos los coeficientes de atención de **gato** en la siguiente frase

Ejemplo:

El gato está en el auto porque se rompió el neumático.

<i>El</i>	<i>0.08</i>
<i>gato</i>	<i>0.35</i>
<i>está</i>	<i>0.15</i>
<i>en</i>	<i>0.05</i>
<i>auto</i>	<i>0.25</i>
<i>neumatico</i>	<i>0.2</i>

Los coeficientes se usan como pesos para **modificar el vector que representa gato**, dándonos una versión contextualizada.

Auto-atención enmascarada

En los modelos que se usan para generar texto de manera autoregresiva pedimos que la atención sea causal o **enmascarada**.

Auto-atención enmascarada

En los modelos que se usan para generar texto de manera autoregresiva pedimos que la atención sea causal o **enmascarada**. Es decir que cada token solo pueda atender a aquellos que vienen antes.

$$A_{\theta}(X_{\leq m-1}, x_m)$$

Ejemplo:

El gato está en el auto porque se rompió el neumático.

*Los coefs. de auto-atención enmascarada de **auto** son:*

<i>El</i>	0.18
<i>gato</i>	0.27
<i>está</i>	0.14
<i>en</i>	0.22
<i>el</i>	0.18
auto	0.0
<i>porque</i>	0.0
<i>se</i>	0.0
<i>rompió</i>	0.0
<i>el</i>	0.0
<i>neumático</i>	0.0

P A R T E II:

Modelos generativos mediante atención (GPT)

Qué es GPT?

La sigla GPT significa Generative Pretrained Transformer. Es un modelo de lenguaje cuyo ingrediente principal es auto-atención ([2018, Sutskever et al. **Improving Language Understanding by Generative Pre-Training**]).

Qué es GPT?

La sigla GPT significa Generative Pretrained Transformer. Es un modelo de lenguaje cuyo ingrediente principal es auto-atención ([2018, Sutskever et al. **Improving Language Understanding by Generative Pre-Training**]).

Este modelo es el ingrediente principal de la mayor parte de chatbots generativos. Es un modelo de lenguaje así que:

$$p(x_m = w_j | x_{[m-\ell, m-1]}) = \mathbb{B} \left(F_{\theta}(x_{[m-\ell, m-1]}) \right)_j$$

con $F_{\theta} : W^{\ell} \rightarrow \mathbb{R}^{|W|}$ una función que depende de parámetros θ .

ahora describiremos la familia de funciones $F_{\theta}(x_{[m-\ell, m-1]})$ de GPT.

- Dado un embedding del contexto

$$X = \begin{pmatrix} | & | & \cdots & | \\ x_1 & x_2 & \cdots & x_\ell \\ | & | & \cdots & | \end{pmatrix}$$

- Calculamos embeddings contextualizados con parámetros compartidos K, Q, V ,

$$Y = \begin{pmatrix} | & | & \cdots & | \\ A_\theta(X_{\leq 0}, x_1) & A_\theta(X_{\leq 1}, x_2) & \cdots & A_\theta(X_{\leq \ell-1}, x_\ell) \\ | & | & \cdots & | \end{pmatrix}$$

Modelo GPT

- Aplicamos renormalización y una fully-connected layer *independientemente* en cada columna.

Modelo GPT

- Aplicamos renormalización y una fully-connected layer *independientemente* en cada columna.

Si G_η es la composición

$$\mathbb{R}^k \xrightarrow{\frac{w}{\|w\|}} \mathbb{R}^k \xrightarrow{D} \mathbb{R}^L \xrightarrow{\sigma} \mathbb{R}^L \xrightarrow{E} \mathbb{R}^k \xrightarrow{\frac{w}{\|w\|}} \mathbb{R}^k$$

con parámetros $\eta := (D, E)$ entonces calculamos

$$Z = \begin{pmatrix} | & | & \cdots & | \\ G_\eta(y_1) & G_\eta(y_2) & \cdots & G_\eta(y_\ell) \\ | & | & \cdots & | \end{pmatrix}$$

Modelo GPT

- Aplicamos renormalización y una fully-connected layer *independientemente* en cada columna.

Si G_η es la composición

$$\mathbb{R}^k \xrightarrow{\frac{w}{\|w\|}} \mathbb{R}^k \xrightarrow{D} \mathbb{R}^L \xrightarrow{\sigma} \mathbb{R}^L \xrightarrow{E} \mathbb{R}^k \xrightarrow{\frac{w}{\|w\|}} \mathbb{R}^k$$

con parámetros $\eta := (D, E)$ entonces calculamos

$$Z = \begin{pmatrix} | & | & \cdots & | \\ G_\eta(y_1) & G_\eta(y_2) & \cdots & G_\eta(y_\ell) \\ | & | & \cdots & | \end{pmatrix}$$

- Se pueden repetir los dos pasos anteriores tantas veces como queramos produciendo un output Z ...

Comentarios sobre la arquitectura:

- Mientras self-attention refleja interacción entre tokens, la composición $E \circ \sigma \circ D$ permite interacciones no lineales entre las componentes de cada embedding contextualizado.

Comentarios sobre la arquitectura:

- Mientras self-attention refleja interacción entre tokens, la composición $E \circ \sigma \circ D$ permite interacciones no lineales entre las componentes de cada embedding contextualizado.
- La utilidad de la composición de capas en modelos de contexto acotado es hacer crecer el tamaño efectivo del contexto.
- El rol de la normalización es evitar que los parámetros crezcan sin control manteniendo todas las capas en esferas.

Modelo GPT

- Finalmente usando la proyección lineal U^T obtenemos distribuciones de probabilidad de la siguiente palabra en cada columna

$$P = \begin{pmatrix} \mathbb{B}(U^T z_1) & \mathbb{B}(U^T z_2) & \cdots & \mathbb{B}(U^T z_\ell) \\ | & | & \cdots & | \\ | & | & \cdots & | \end{pmatrix}$$

Modelo GPT

- Finalmente usando la proyección lineal U^T obtenemos distribuciones de probabilidad de la siguiente palabra en cada columna

$$P = \begin{pmatrix} \mathbb{B}(U^T z_1) & \mathbb{B}(U^T z_2) & \dots & \mathbb{B}(U^T z_\ell) \\ | & | & \dots & | \\ | & | & \dots & | \end{pmatrix}$$

El procedimiento preserva causalidad y puede realizarse en paralelo. Cada contexto aporta ℓ términos a la función objetivo.

Modelo GPT

- Finalmente usando la proyección lineal U^T obtenemos distribuciones de probabilidad de la siguiente palabra en cada columna

$$P = \begin{pmatrix} \mathbb{B}(U^T z_1) & \mathbb{B}(U^T z_2) & \cdots & \mathbb{B}(U^T z_\ell) \\ | & | & \cdots & | \\ | & | & \cdots & | \end{pmatrix}$$

El procedimiento preserva causalidad y puede realizarse en paralelo. Cada contexto aporta ℓ términos a la función objetivo.

Para generar nuevo texto sampleamos de manera autoregresiva, de acuerdo a

$$p(x_{\ell+1} = w_j | x_{[1,\ell]}) = \mathbb{B}(U^T z_\ell)_j$$

En la descripción anterior faltan tres cosas:

- 1 **Positional embedding.** La función de atención es independiente del orden de columnas así que enriquecemos las columnas con información adicional sobre su posición relativa.
- 2 **Multihead attention.** Tenemos varias funciones de atención con parámetros $\theta_h := (Q^h, K^h, V^h)$ y las combinamos

$$A_{\theta}(X, w) = \sum_{h=1}^H W^h(A_{\theta_h}(X, w))$$

donde las matrices $W^h \in \mathbb{R}^h \times \mathbb{R}^k$ son parámetros adicionales.

- 3 **Skip connections.** Típicamente el output de cada capa no es $Z^{(j)}$ sino la suma de los Z 's y Y 's de bloques anteriores.

Parámetros y datos de entrenamiento

La descripción anterior nos deja muchos parámetros libres: Qué dimensiones deben tener las matrices? Cuántas capas de atención?

Year	Model	Number of Parameters	Training Data Size
2018	GPT (GPT-1)	117 million	5 GB (BooksCorpus)
2019	GPT-2	1.5 billion	40 GB (WebText)
2020	GPT-3	175 billion	570 GB (Common Crawl, WebText2, Books, Wikipedia)
2022	GPT-3.5	175 billion	570 GB (Similar to GPT-3 with further refinements)
2023	GPT-4 (Speculative)	Estimated 1 trillion+	Multi-terabyte

Parámetros y datos de entrenamiento

La descripción anterior nos deja muchos parámetros libres: Qué dimensiones deben tener las matrices? Cuántas capas de atención?

Year	Model	Number of Parameters	Training Data Size
2018	GPT (GPT-1)	117 million	5 GB (BooksCorpus)
2019	GPT-2	1.5 billion	40 GB (WebText)
2020	GPT-3	175 billion	570 GB (Common Crawl, WebText2, Books, Wikipedia)
2022	GPT-3.5	175 billion	570 GB (Similar to GPT-3 with further refinements)
2023	GPT-4 (Speculative)	Estimated 1 trillion+	Multi-terabyte

El entrenamiento de GPT4 requirió 12-64 GWh (GigaWatthours):

Parámetros y datos de entrenamiento

La descripción anterior nos deja muchos parámetros libres: Qué dimensiones deben tener las matrices? Cuántas capas de atención?

Year	Model	Number of Parameters	Training Data Size
2018	GPT (GPT-1)	117 million	5 GB (BooksCorpus)
2019	GPT-2	1.5 billion	40 GB (WebText)
2020	GPT-3	175 billion	570 GB (Common Crawl, WebText2, Books, Wikipedia)
2022	GPT-3.5	175 billion	570 GB (Similar to GPT-3 with further refinements)
2023	GPT-4 (Speculative)	Estimated 1 trillion+	Multi-terabyte

El entrenamiento de GPT4 requirió 12-64 GWh (GigaWatthours):

- *La electricidad TOTAL gastada en Uruguay durante dos días.*
- *Cuesta entre 50 y 100 millones de dólares.*

Small Language models

En *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* **[Abril 2023, R.Eldan y Y. Li]** muestran que hay otro camino posible.

Small Language models

En *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* **[Abril 2023, R.Eldan y Y. Li]** muestran que hay otro camino posible.

La arquitectura de GPT es capaz de aprender el lenguaje con coherencia gramatical y semántica con un dataset mucho más pequeño si este dataset es **suficientemente diverso y de buena calidad**.

Small Language models

En *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* [Abril 2023, R.Eldan y Y. Li] muestran que hay otro camino posible.

La arquitectura de GPT es capaz de aprender el lenguaje con coherencia gramatical y semántica con un dataset mucho más pequeño si este dataset es **suficientemente diverso y de buena calidad**.

Eldan y Li crean, usando GPT-3.5 y GPT-4 un dataset sintético (TinyStories) de historias para niños (15000 historias de 200 palabras c/u $\sim 30MB$).

Small Language models

En *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* [Abril 2023, R.Eldan y Y. Li] muestran que hay otro camino posible.

La arquitectura de GPT es capaz de aprender el lenguaje con coherencia gramatical y semántica con un dataset mucho más pequeño si este dataset es **suficientemente diverso y de buena calidad**.

Eldan y Li crean, usando GPT-3.5 y GPT-4 un dataset sintético (TinyStories) de historias para niños (15000 historias de 200 palabras c/u $\sim 30MB$).

Usando 28mm de parámetros pueden crear un modelo de lenguaje que genera mejores historias que GPT-2XL (1.5B).

- *Junto con B. Arnuti reproducimos los resultados de [Eldan,Li] y construimos un GPT que tiene aún mejor performance (GPTTinyStories).*

- *Junto con B. Arnuti reproducimos los resultados de [Eldan,Li] y construimos un GPT que tiene aún mejor performance (GPTTinyStories).*
- *El entrenamiento tomó 10 horas en una Desktop (RTX 4070TI de 12GB de VRAM).*

- *Junto con B. Arnuti reproducimos los resultados de [Eldan,Li] y construimos un GPT que tiene aún mejor performance (GPTTinyStories).*
- *El entrenamiento tomó 10 horas en una Desktop (RTX 4070TI de 12GB de VRAM).*

- *Junto con B. Arnuti reproducimos los resultados de [Eldan,Li] y construimos un GPT que tiene aún mejor performance (GPTTinyStories).*
- *El entrenamiento tomó 10 horas en una Desktop (RTX 4070TI de 12GB de VRAM).*

Damos una frase inicial a GPTTinyStories:

One day, her mom asked her to pick some vegetables from the garden. Lily was excited to help. As Lily picked the vegetables, she saw a dangerous snake on the ground. She was scared and called out to her mom. "Mom, there is a dangerous snake" y dejamos que genere libremente de ahí en adelante...

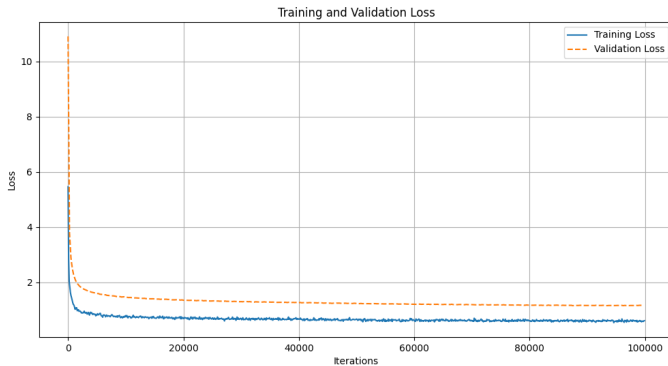
1000 Iterations: One day, her mom asked her to pick some vegetables from the garden. Lily was excited to help. As Lily picked the vegetables, she saw a dangerous snake on the ground. She was scared and called out to her mom. "Mom, there is a dangerous snake" We are you, but I'm sorry!" Once upon a time, there was a little boy named Lily. They went to work with her mom! They looked at the park. They found a big tree. They played with the garden. He would be sad.

5000 Iterations: One day, her mom asked her to pick some vegetables from the garden. Lily was excited to help. As Lily picked the vegetables, she saw a dangerous snake on the ground. She was scared and called out to her mom. "Mom, there is a dangerous snake" sound. She ran to her mom and said, "Mom, I want to observe the bees. She is scared of the snake. She does not like to fall. She follows her mom. One sunny day, they decided to visit the tiger. They walked to see the tiger, the tiger, and the mouse. They walked with big eyes and a big smile. The tiger smiled at them and said, "Thank you, little boy. You are very kind. I am very kind."

10000 Iterations: One day, her mom asked her to pick some vegetables from the garden. Lily was excited to help. As Lily picked the vegetables, she saw a dangerous snake on the ground. She was scared and called out to her mom. "Mom, there is a dangerous snake" her mom said. Lily's mom came outside and saw the dangerous snake. She called to Lily, "Lily, we need to help! Come here, I will help you." Lily and her mom took the dangerous snake to the house. They gave it food and water.

Our model, final 100000 Iterations: One day, her mom asked her to pick some vegetables from the garden. Lily was excited to help. As Lily picked the vegetables, she saw a dangerous snake on the ground. She was scared and called out to her mom. "Mom, there is a dangerous snake" she said. Her mom came outside and saw the snake. She was scared too, but she knew she had to help her little girl. "Don't worry, Lily. I will get rid of it," her mom said. Her mom grabbed a big stick and waved it at the snake. The snake didn't like the stick, and it got scared and went away. Lily was happy that she could help her mom with the vegetables. She gave her mom a big hug and said, "We did it together!" Her mom smiled and said, "Thank you, Lily. You are a great helper!"

Resultados



<https://huggingface.co/spaces/Counterplan/TinyTales>

P A R T E III:

Cómo construir un modelo de lenguaje?

Para qué construir modelos de lenguaje (SLM)?

Hay varias razones por las que esto es una idea interesante:

- ❶ Para muchas aplicaciones las capacidades de un LLM son excesivas y difíciles de controlar.
- ❷ Los costos (energéticos y monetarios) de correr LLMs son enormes y están fuera de nuestro control.
- ❸ Al usar un LLM típicamente se pierde la soberanía de los datos.

DISCUSSION

Cómo construir modelos pequeños de lenguaje (SLMs)?

A este respecto hay varias ideas.

Cómo construir modelos pequeños de lenguaje (SLMs)?

A este respecto hay varias ideas. Todas involucran el uso de grandes modelos de lenguaje LLMs como apoyo en la construcción de SLMs.

Concretamente es posible construirlos a partir de **datos (artificiales) de buena calidad** o a partir de las **distribuciones de probabilidad de un modelo entrenado** (model distillation).

Cómo construir modelos pequeños de lenguaje (SLMs)?

A partir de **datos de buena calidad**.

- Es posible **generar** datos de buena calidad de manera artificial

Cómo construir modelos pequeños de lenguaje (SLMs)?

A partir de **datos de buena calidad**.

- Es posible **generar** datos de buena calidad de manera artificial
- Es posible **evaluar** la calidad del aprendizaje de manera artificial

Analizaremos estas ideas en un caso concreto siguiendo el trabajo de Ronen Eldan y Yuanzhi Li (Microsoft research) quienes inventaron esta metodología [**Eldan y Li, 2023**].

Objetivo: *Construir un corpus de entrenamiento que combine los elementos cualitativos presentes en el lenguaje natural, como la gramática, el vocabulario y algo de razonamiento, pero que también sea más pequeño, menos diverso y más restringido en cuanto a su contenido.*

Cómo construir modelos pequeños de lenguaje (SLMs)?

La otra idea es **model distillation**. Asumiendo que la información a nuestra disposición consiste de **un modelo completamente entrenado y de un conjunto de datos**.

Cómo construir modelos pequeños de lenguaje (SLMs)?

La otra idea es **model distillation**. Asumiendo que la información a nuestra disposición consiste de **un modelo completamente entrenado y de un conjunto de datos**.

El principal desafío al usar LLMs para generar datos de entrenamiento es crear un conjunto de datos que sea lo suficientemente **diverso**: pedirle a estos modelos que generen historias, incluso si se establece una temperatura alta en la generación, seguirá produciendo un conjunto de datos muy repetitivo, cuya diversidad está muy lejos de lo que se requiere para entrenar un modelo de lenguaje.

Cómo construir modelos pequeños de lenguaje (SLMs)?

Para abordar el problema de crear un conjunto de datos diverso, recopilamos un vocabulario compuesto por aproximadamente 1500 palabras básicas, que intentan imitar el vocabulario típico de un niño de 3 a 4 años, separadas en sustantivos, verbos y adjetivos.

Cómo construir modelos pequeños de lenguaje (SLMs)?

Para abordar el problema de crear un conjunto de datos diverso, recopilamos un vocabulario compuesto por aproximadamente 1500 palabras básicas, que intentan imitar el vocabulario típico de un niño de 3 a 4 años, separadas en sustantivos, verbos y adjetivos.

En cada generación, se eligen al azar 3 palabras (un verbo, un sustantivo y un adjetivo) y una lista de características que una historia podría tener (como un diálogo, un giro en la trama, un final triste o una moraleja). Se le indica al LLM que genere una historia que combine de algún modo estas palabras y características en la narración.

Cómo construir modelos pequeños de lenguaje (SLMs)?

Para abordar el problema de crear un conjunto de datos diverso, recopilamos un vocabulario compuesto por aproximadamente 1500 palabras básicas, que intentan imitar el vocabulario típico de un niño de 3 a 4 años, separadas en sustantivos, verbos y adjetivos.

En cada generación, se eligen al azar 3 palabras (un verbo, un sustantivo y un adjetivo) y una lista de características que una historia podría tener (como un diálogo, un giro en la trama, un final triste o una moraleja). Se le indica al LLM que genere una historia que combine de algún modo estas palabras y características en la narración.

Como veremos más adelante, esto aumenta considerablemente la diversidad del conjunto de datos, obligando a que las historias abarquen todo el vocabulario con el que un niño está familiarizado, e incluyan una variedad rica de formas de combinar diferentes conceptos.

Prompting para historias

Write a short story (3-5 paragraphs) which only uses very simple words that a 3 year old child would likely understand. The story should use the verb "decorate", the noun "thunder" and the adjective "ancient". The story should have the following features: the story should contain at least one dialogue, the story has a bad ending. Remember to only use simple words!

Prompting para historias

Write a short story (3-5 paragraphs) which only uses very simple words that a 3 year old child would likely understand. The story should use the verb "decorate", the noun "thunder" and the adjective "ancient". The story should have the following features: the story should contain at least one dialogue, the story has a bad ending. Remember to only use simple words!

Once upon a time, in an ancient house, there lived a girl named Lily. She loved to decorate her room with pretty things. One day, she found a big box in the attic. She opened it and saw many shiny decorations. Lily was very happy and decided to use them in her room. As Lily was decorating her room, the sky outside became dark. There was a loud thunder sound, and Lily got scared. She ran to her mom and said, "Mommy, the thunder is so loud!" Her mom hugged her and said, "Don't worry, it will pass soon." But the thunder did not stop. It got louder and louder, and the ancient house started to shake. Suddenly, the roof fell down on the pretty decorations. Lily was sad because her room was not pretty anymore. The end.

Prompting para historias

El corpus generado, llamado **TinyStories**, consiste de 2.5 millones de historias y mide 1GB. Está públicamente disponible.

Mecanismos como este permitirían generar corpora pequeños para tareas específicas.

Cómo evaluar la calidad de un modelo de lenguaje?

Cómo evaluar la calidad de un modelo de lenguaje?

Metodología GPT-Eval

Cómo evaluar la calidad de un modelo de lenguaje?

Metodología GPT-Eval

Proporcionamos al modelo el inicio de una historia (manualmente preparada), y generamos una continuación usando el modelo. Proporcionamos el comienzo de la historia junto con la continuación generada a GPT-4, pidiéndole que califique la tarea de continuación en términos de **gramática**, **creatividad** y **coherencia** con el inicio de la historia. El inicio de la historia suelen contener una oración parcial que necesita ser completada e instruimos a GPT-4 para que ponga especial atención acerca de qué tan bien se completa. Procuramos redactarlas de manera que completar la oración truncada fuera una tarea gramaticalmente no trivial.

Prompt de evaluacion

In the following exercise, the student is given a beginning of a story. The student needs to complete it into a full story. The exercise tests the students language abilities and creativity. The symbol * marks the separator between the prescribed beginning and the student's completion:**

Once upon a time, in an ancient house, there lived a girl named Lily. She loved to decorate her room with pretty things. One day, she found a big box in the attic. She opened it and saw many shiny decorations. Lily was very happy and decided to use them in her room. As Lily was decorating her room, the sky outside became dark. There was a loud*** thunderclap and a bright flash of lightning. Lily was a little scared, but she knew she had to be brave. She told her mom and dad, and they all went outside to see what was going on. When they got outside, they saw a big storm coming. The wind was blowing hard, and the rain was pouring down. Lily, her mom, and her dad knew they had to find a safe place to hide. They found a big tree and hid under it. The storm passed, and the sun came out again. Lily, her mom, and her dad were all safe and warm inside their ancient house.

Please provide your general assessment about the part written by the student (the one after the * symbol). Is it gramatically correct? Is it consistent with the beginning of the story? Pay special attention to whether the student manages to complete the sentence which is split in the middle by the separator ***.**

Prompt de evaluación

GPT: The student's completion of the story is mostly consistent with the beginning of the story. It maintains the focus on Lily and her family, and the sentence split by the separator is completed correctly. However, the student's addition does not fully integrate the shiny decorations found in the attic, which were a significant part of the beginning. The grammar is generally correct, but there are a few minor errors: `<list omitted>`. Overall, the student's completion of the story demonstrates adequate language abilities and creativity, but could benefit from better integration of the shiny decorations and minor grammar improvements.

Prompt de evaluacion

GPT: The student's completion of the story is mostly consistent with the beginning of the story. It maintains the focus on Lily and her family, and the sentence split by the separator is completed correctly. However, the student's addition does not fully integrate the shiny decorations found in the attic, which were a significant part of the beginning. The grammar is generally correct, but there are a few minor errors: `<list omitted>`. Overall, the student's completion of the story demonstrates adequate language abilities and creativity, but could benefit from better integration of the shiny decorations and minor grammar improvements.

Now, grade the student's completion in terms of grammar, creativity, consistency with the story's beginning and whether the plot makes sense. Moreover, please provide your best guess of what the age of the student might be, as reflected from the completion. Choose from possible age groups: A: 3 or under. B: 4-5. C: 6-7. D: 8-9. E: 10-12. F: 13-16.

GPT: Grammar: 8/10, Creativity: 7/10, Consistency: 7/10, Age group: E

La capacidad de analizar de manera sistemática enormes cantidades de texto mediante GPT-Eval permite usar técnicas estadísticas para la evaluación de la calidad de un modelo entrenado y para comparar entre alternativas. Es una metodología muy valiosa y aplicable generalmente.

Model Distillation

[Hinton et al, 2015] Many insects have a larval form that is optimized for extracting energy and nutrients from the environment and a completely different adult form that is optimized for the very different requirements of traveling and reproduction. In large-scale machine learning, we typically use very similar models for the training stage and the deployment stage despite their very different requirements: For tasks like speech and object recognition, training must extract structure from very large, highly redundant datasets but it does not need to operate in real time and it can use a huge amount of computation. Deployment to a large number of users, however, has much more stringent requirements on latency and computational resources. The analogy with insects suggests that we should be willing to train very cumbersome models if that makes it easier to extract structure from the data. The cumbersome model could be an ensemble of separately trained models or a single very large model. Once the cumbersome model has been trained, we can then use a different kind of training, which we call **“distillation”** to **transfer the knowledge from the cumbersome model to a small model** that is more suitable for deployment.

Cómo destilar modelos pequeños de lenguaje (SLMs)?

*Mas concretamente tenemos la distribución de probabilidad del modelo **profesor** p_T que podemos evaluar*

$$p_T(x_m | x_{[m-1-\ell, m-1]}) = \mathbb{B}(F_{\theta_T}(x_{[m-\ell, m-1]}))$$

*en cualquier contexto y queremos entrenar un modelo **estudiante** p_S con un numero mucho menor de parámetros θ_S .*

Adicionalmente tenemos un dataset $d_1 d_2 \dots d_M$.

Cómo destilar modelos pequeños de lenguaje (SLMs)?

Queremos que el modelo estudiante produzca **distribuciones de probabilidad parecidas** a las generadas por el modelo instructor.

Cómo destilar modelos pequeños de lenguaje (SLMs)?

Queremos que el modelo estudiante produzca **distribuciones de probabilidad parecidas** a las generadas por el modelo instructor. Para tal efecto queremos minimizar la divergencia de Kullback-Liebler entre ellos

$$D(p_T|p_S) := \sum_{w \in W} p_T(w|d_{[j-1-\ell, j-1]}) \log \left(\frac{p_T(w|d_{[j-1-\ell, j-1]})}{p_S(w|d_{[j-1-\ell, j-1]})} \right)$$

Cómo destilar modelos pequeños de lenguaje (SLMs)?

Queremos que el modelo estudiante produzca **distribuciones de probabilidad parecidas** a las generadas por el modelo instructor. Para tal efecto queremos minimizar la divergencia de Kullback-Liebler entre ellos

$$D(p_T|p_S) := \sum_{w \in W} p_T(w|d_{[j-1-\ell, j-1]}) \log \left(\frac{p_T(w|d_{[j-1-\ell, j-1]})}{p_S(w|d_{[j-1-\ell, j-1]})} \right)$$

*La divergencia de Kullback Liebler mide la **sorpesa promedio** que tiene el instructor (T) al ver la distribución propuesta por el estudiante (S).*

Cómo destilar modelos pequeños de lenguaje (SLMs)?

Queremos que el modelo estudiante produzca **distribuciones de probabilidad parecidas** a las generadas por el modelo instructor. Para tal efecto queremos minimizar la divergencia de Kullback-Liebler entre ellos

$$D(p_T|p_S) := \sum_{w \in W} p_T(w|d_{[j-1-\ell, j-1]}) \log \left(\frac{p_T(w|d_{[j-1-\ell, j-1]})}{p_S(w|d_{[j-1-\ell, j-1]})} \right)$$

*La divergencia de Kullback Liebler mide la **sorpesa promedio** que tiene el instructor (T) al ver la distribución propuesta por el estudiante (S).*

Mas precisamente, se minimiza (en θ_S) el promedio de las divergencias de Kullback-Liebler a lo largo de todo el set de entrenamiento

$$\frac{1}{N} \sum_{j=1}^N D_j(p_T|p_S)(\theta) = \frac{1}{N} \sum_{j=1}^N \left(\sum_{w \in W} p_T(w|d_{[j-1-\ell, j-1]}) \log \left(\frac{p_T(w|d_{[j-1-\ell, j-1]})}{p_S(w|d_{[j-1-\ell, j-1]})} \right) \right)$$

Cómo destilar modelos pequeños de lenguaje (SLMs)?

Adicionalmente se complementa el aprendizaje del modelo estudiante adicionando al objetivo un término de máxima verosimilitud mejorando su capacidad de generar los datos de entrenamiento

$$-\sum_{j=1}^N \log p_S(d_j | d_{[j-1-\ell, j-1]})$$

Cómo destilar modelos pequeños de lenguaje (SLMs)?

Adicionalmente se complementa el aprendizaje del modelo estudiante adicionando al objetivo un término de maxima verosimilitud mejorando su capacidad de generar los datos de entrenamiento

$$-\sum_{j=1}^N \log p_S(d_j | d_{[j-1-\ell, j-1]})$$

Por último, es común **aumentar la temperatura** M (tanto en T como en S) para obtener distribuciones más uniformes (la idea acá es que en modelos entrenados se observan enormes diferencias entre probabilidades muy pequeñas y crecer M las magnifica).

$$q_i = \frac{\exp(\frac{z_i}{M})}{\sum_j \exp(\frac{z_j}{M})}$$

Cuando $M = 1$ da la distribucion de Boltzmann usual, pero para $M \gg 1$ acerca a la uniforme.

Distillation es una tecnica fundamental en IA para la construcción de modelos más pequeños y más baratos.