Pontificia Universidad JAVERIANA Cali

Encuentro Sincrónico 6

Procesamiento de texto y Transformers

Pontificia Universidad JAVERIANA Cali

Table of Contents

Procesamiento de texto

- Como se ha estudiado previamente, los modelos secuenciales son usados en diversas tareas, incluyendo:
  - Traducción automática.
  - Generación de texto.
  - Análisis de sentimientos.
- Estas tareas implican trabajar con datos textuales.
- Sin embargo, los sistemas de cómputo no tienen la capacidad de entender palabras.
- En ese sentido, es necesario aplicar una serie de pasos con el fin de codificar los textos a un lenguaje que entiendan los computadores.

Procesamiento de texto: Estandarización y tokenización

- Proceso que consiste en garantizar homogeneidad en la forma y puntuación de los textos (Mayúsculas, sin puntuación).

  El gato, que estaba sentado, no es nuestro. $\rightarrow$ el gato que estaba sentado no es nuestro

- Por su lado, la tokenización consiste en dividir los textos en unidades de análisis, la división más común es en palabras; sin embargo, puede hacerse a nivel de caracteres (letras) o conjuntos de palabras.

  El gato, que estaba sentado, no es nuestro. $\rightarrow$ ['el', 'gato', 'que', 'estaba', 'sentado', 'no', 'es', 'nuestro']

Procesamiento de texto: Representación Rala

- Ahora, se debe darle una codificación numérica a cada token.
- Existen diversas formas, una de ellas es una representación rala (sparse), la cual es similar a la codificación de las variables categóricas (one-hot encoding).

	el	gato	que	estaba	sentado	no	es	nuestro
gato:	0	1	0	0	0	0	0	0
sentado:	0	0	0	0	1	0	0	0

Procesamiento de texto: Representación Rala

- Como se nota, la representación para cada palabra necesitaría un vector con muchas posiciones (tantas como palabras en el idioma).
- Lo anterior puede llevar a problemas de memoria y cantidad de parámetros.
- Una alternativa es definir un diccionario reducido, con las palabras más comunes del idioma.
- Sin embargo, esto conlleva a otro problema. Suponer un cambio en la oración que se ha usado como ejemplo.

El ornitorrinco, que estaba sentado, no es nuestro.

Procesamiento de texto: Representación Rala

- La palabra ornitorrinco no es muy común en el idioma Español, así que es posible que no esté dentro del diccionario elegido. ¿Qué se puede hacer?

- Debemos incluir un nuevo elemento ([UNK]) en el diccionario para codificar todas las palabras que no están en el diccionario.

	el	gato	que	estaba	sentado	no	es	nuestro	[UNK]
el:	1	0	0	0	0	0	0	0	0
Ornitorrinco:	0	0	0	0	0	0	0	0	1

Procesamiento de texto: Embebimientos de texto (Words embeddings)

- La representación rala necesita vectores con muchas dimensiones para representar una palabra, donde todas las entradas, menos una, son ceros.

- Evidentemente no es una representación muy efectiva. Una alternativa es usar una representación vectorial de las palabras.

- Por ejemplo, estas pueden ser las representaciones de las palabras “Python” y “Redes”

$$\text{Python} = \begin{bmatrix} 0{,}1 \\ 0{,}6 \\ -0{,}34 \\ 0{,}01 \end{bmatrix} \qquad \text{Redes} = \begin{bmatrix} 0{,}2 \\ 0{,}4 \\ 0{,}9 \\ 0{,}1 \end{bmatrix}$$

Procesamiento de texto: Embebimientos de texto (Words embeddings)

- Ahora la pregunta es ¿Cómo estimar los vectores que representan las palabras?
- Esa repuesta se puede contestar a partir de modelos de redes neuronales (fully-connected).
- La idea es plantear un problema falso donde se tiene que predecir una palabra a partir de su contexto (fill blank).
- Por ejemplo, suponer la orción: “El gato estaba”, la idea es configurar una red que prediga la palabra “gato” a partir de las palabras cercanas, en este caso “el” y “estaba”

Procesamiento de texto: Embebimientos de texto (Words embeddings)

Cada palabra se se representa de forma rala (one-hot encoding)

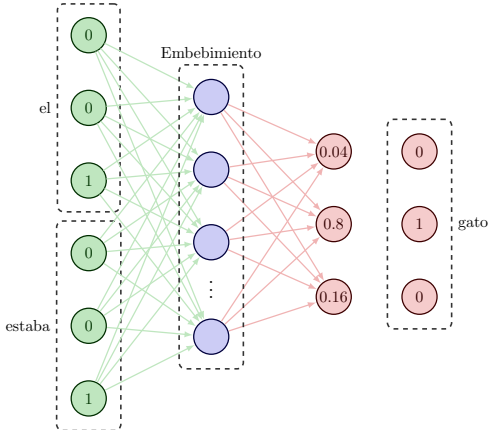
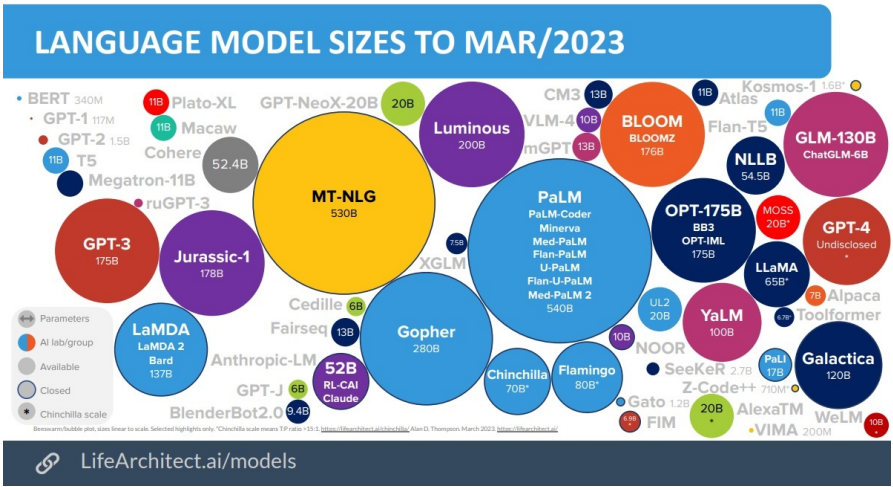


Table of Contents

Transformers: LLM

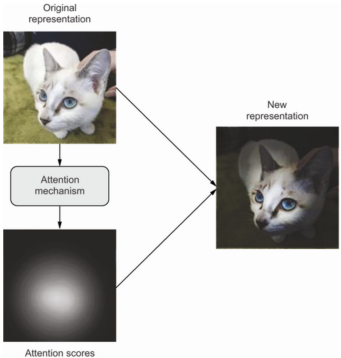
- La arquitectura Transformers fue presentada en Junio de 2017. Su concepción original fue en tareas de traducción automática. Sin embargo, esta arquitectura ha promovido diferentes modelo de lenguaje, incluyendo:
  - GPT: Primero modelo Transformer pre-entrenado, usado para fine-tuning en varias tareas de Procesamiento de lenguaje natural.
  - Bert: Modelo pre-entrenado, diseñado para producir mejores resúmenes de las frases.
  - GPT-2: Una versión mejorada y más grande que GPT.
  - GPT-3: una versión mucho más grande que GPT-2.

Transformers: LLM

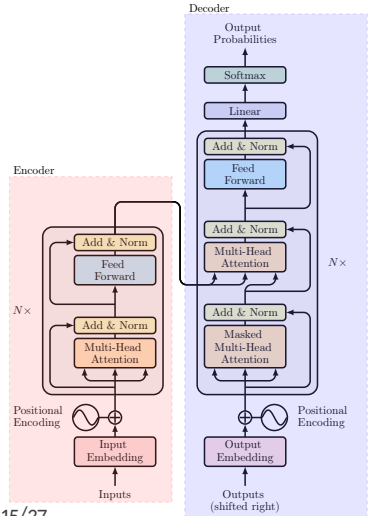


Transformers: Arquitectura Original

- El artículo Attention is all you need presentó la primera versión de una arquitectura transformer, la cual está formada únicamente a través de mecanismos de atención.

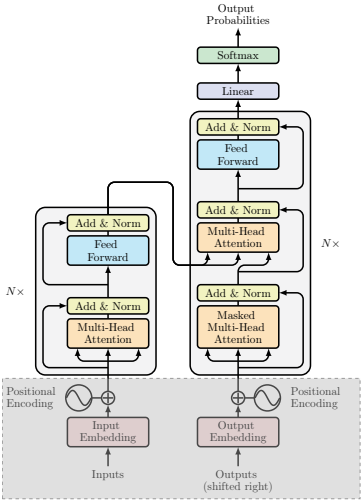


Transformers: Arquitectura Original



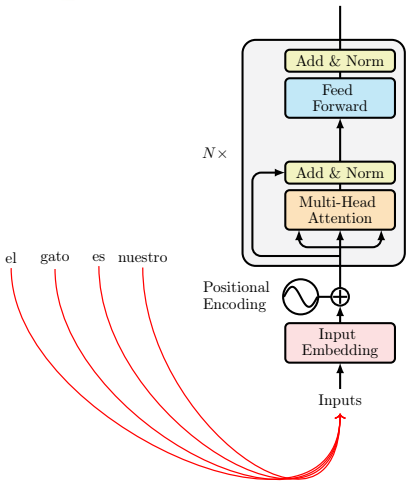
- Como se dijo previamente, la arquitectura original se enfocaba en resolver tareas de traducción automática.
- Se puede observar que no existe ningún tipo de arquitectura como las redes secuenciales o convolucionales. Únicamente mecanismos de atención.
- Así, se tienen dos grandes componentes: El encoder y el decoder.
- El encoder codifica la información de las frases en el idioma base.
- El decoder toma esa codificación más la representación de las frases traducidas.

Transformers: Codificación posicional



- Se empieza el análisis de los transformers estudiando las entradas de tanto el encoder como el decoder.
- La primera parte consiste en codificar los tokens a partir de un embebimiento como el que se vio previamente.
- Ahora, como segundo elemento en la arquitectura se encuentran la codificación posicional.

Transformers: Codificación posicional



- La codificación posicional es similar para el encoder como el decoder. Para este caso se elige el encoder.
- En un modelo secuencial como los vistos en sesiones anteriores, las entradas se analizan de forma secuencial.
- Esto puede ser problemático ya que no permitiría la paralelización, lo cual ralentiza el procesamiento.
- Así, la idea es que todas las palabras ingresen al mismo momento; sin embargo, recordar que el orden de las palabras es vital en este tipo de datos.

Transformers: Codificación posicional

- La solución planteada por los transformers es asignarle una codificación a las posición de cada palabra.
- La forma más simple es sumarle la posición a la codificación de cada palabra.
- Sin embargo, no es la mejor opción dado que la codificación de la posición predomina sobre la codificación del embebimiento.

$$\underset{\text{el}}{\begin{bmatrix} 0 \\ 0{,}1 \\ 0{,}3 \\ 0{,}4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{\text{gato}}{\begin{bmatrix} 0{,}6 \\ 0{,}9 \\ -0{,}1 \\ 0{,}01 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}} \quad \underset{\text{es}}{\begin{bmatrix} 0{,}54 \\ 0{,}32 \\ 0{,}33 \\ -0{,}9 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}} \quad \underset{\text{nuestro}}{\begin{bmatrix} 0{,}6 \\ 0{,}79 \\ -0{,}43 \\ 0{,}9 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}}$$

Transformers: Codificación posicional

- Una alternativa es crear una codificación que resulta de dividir el índice de cada palabra por la longitud total de la secuencia.
- Sin embargo, las secuencias varían en su longitud.

Original	0	1	2	3
Escalado	0	0.25	0.5	0.75

el

$$\begin{bmatrix} 0 \\ 0{,}1 \\ 0{,}3 \\ 0{,}4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

gato

$$\begin{bmatrix} 0{,}6 \\ 0{,}9 \\ -0{,}1 \\ 0{,}01 \end{bmatrix} + \begin{bmatrix} 0{,}25 \\ 0{,}25 \\ 0{,}25 \\ 0{,}25 \end{bmatrix}$$

es

$$\begin{bmatrix} 0{,}54 \\ 0{,}32 \\ 0{,}33 \\ -0{,}9 \end{bmatrix} + \begin{bmatrix} 0{,}5 \\ 0{,}5 \\ 0{,}5 \\ 0{,}5 \end{bmatrix}$$

nuestro

$$\begin{bmatrix} 0{,}6 \\ 0{,}79 \\ -0{,}43 \\ 0{,}9 \end{bmatrix} + \begin{bmatrix} 0{,}75 \\ 0{,}75 \\ 0{,}75 \\ 0{,}75 \end{bmatrix}$$

Transformers: Codificación posicional

- En resumen, se requiere de una codificación que no tenga número grandes y que no dependa del conocimiento de la longitud de las secuencias.
- Los transformers emplean las funciones seno y coseno para esta codificación

$$\sin\left(\frac{pos}{1000^{\frac{2j}{d}}}\right)$$
$$\cos\left(\frac{pos}{1000^{\frac{2j}{d}}}\right),$$

donde pos indica la posición de la palabra dentro de la secuencia, d es el tamaño del embebimiento y j es un indicador.

Transformers: Codificación posicional

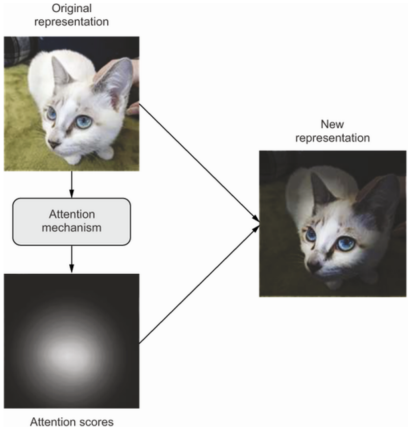
el

$$\begin{bmatrix} 0 \\ 0{,}1 \\ 0{,}3 \\ 0{,}4 \end{bmatrix} + \begin{bmatrix} \sin\left(\frac{0}{1000^{\frac{2*0}{4}}}\right) \\ \cos\left(\frac{0}{1000^{\frac{2*0}{4}}}\right) \\ \sin\left(\frac{0}{1000^{\frac{2*1}{2}}}\right) \\ \cos\left(\frac{0}{1000^{\frac{2*1}{4}}}\right) \end{bmatrix}$$

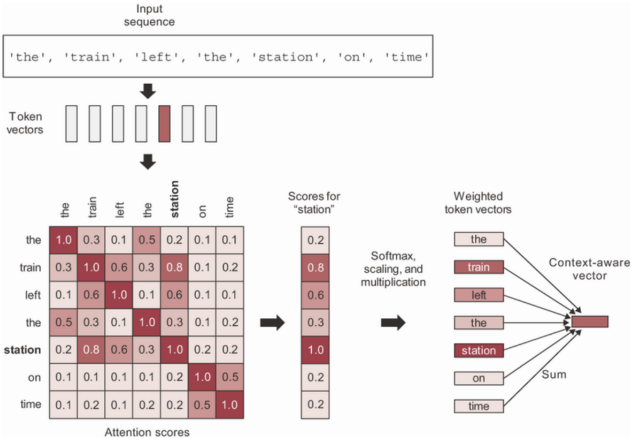
gato

$$\begin{bmatrix} 0{,}6 \\ 0{,}9 \\ -0{,}1 \\ 0{,}01 \end{bmatrix} + \begin{bmatrix} \sin\left(\frac{1}{1000^{\frac{2*0}{4}}}\right) \\ \cos\left(\frac{1}{1000^{\frac{2*0}{4}}}\right) \\ \sin\left(\frac{1}{1000^{\frac{2*1}{2}}}\right) \\ \cos\left(\frac{1}{1000^{\frac{2*1}{4}}}\right) \end{bmatrix}$$

Transformers: Attention



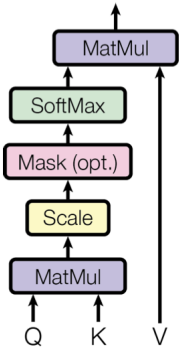
Transformers: Self-Attention[1]

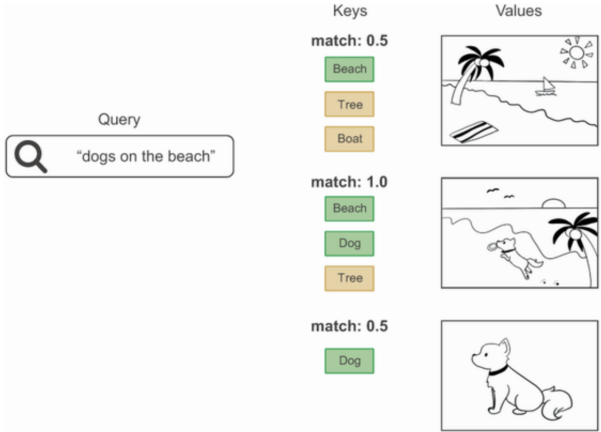


[1]Deep Learning with Python, Second Edition

Transformers: Self-Attention[2]

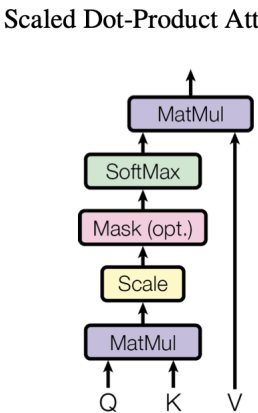
Scaled Dot-Product Attention





[2]Deep Learning with Python, Second Edition, https://arxiv.org/pdf/1706.03762.pdf

Transformers: Masked Self-Attention[3]

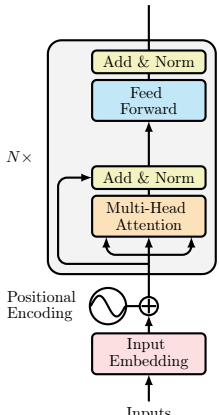


- El mecanismo de atención con máscara pretende "eliminar" algunas componentes. Por ejemplo, los valores que se usan para garantizar que todas las secuencias tengan las misma longitud.
- En la matriz de valores de atención, los valores a enmascarar se fijan en $-\infty$, de tal forma que al pasar por la función Softmax tiendan a cero.

$$\begin{bmatrix} 1 & 0{,}2 & 0{,}6 \\ 0{,}1 & 1 & 0{,}8 \\ 0{,}2 & 0{,}1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -\infty & -\infty \\ 0{,}1 & 1 & -\infty \\ 0{,}2 & 0{,}1 & 1 \end{bmatrix}$$

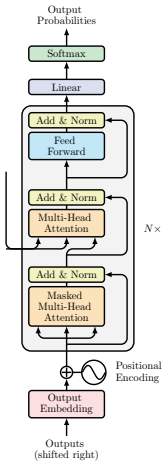
[3]Deep Learning with Python, Second Edition, https://arxiv.org/pdf/1706.03762.pdf

Transformers: Encoder



- La idea de los encoders es extraer características de las entradas (En otras palabras representar las entradas).
- Estas representaciones pueden ser usadas de diferentes maneras. Por ejemplo, si a la salida del encoder se usa una fully connected, se puede usar para clasificación de textos (Análisis de sentimientos).
- Por otro lado, la salida del encoder puede usarse en un decoder con el fin de construir aplicaciones de traducción automática o generación de texto.

Transformers: Decoder



- El decoder toma las características extraídas en el encoder para realizar tareas como generación de texto o traducción.
- El decoder presta “atención” a las palabras más adecuadas para la traducción.

  “You like this course”
- En la traducción automática el decoder tiene acceso a todas las palabras pasadas, no las futuras. Para omitir las futuras se usa el mecanismo de atención con máscara.
- La segunda capa de atención se denomina cruzada ya que Query (Q) y Key (K) provienen del encoder.