Introducción a los mecanismos de atención entranformers

Víctor Germán Mijangos de la Cruz

Departamento de Matemáticas
Facultad de Ciencias, UNAM

Introducción

Modelos secuenciales

Modelos secuenciales

Una red neuronal para secuencias toma como entrada un conjunto de datos secuenciales $x^{(1)}x^{(2)}\cdots x^{(T)}$ y cuya salida se estima como:

$$f(x^{(1)}x^{(2)}\cdots x^{(T)}) = \phi\big(W^{(out)}h^{(1:T)} + b^{(out)}\big)$$

donde $h^{(1:t)}$ es una representación profunda de los datos de entrada, ϕ es la función de activación en la salida y $W^{(out)}$ y $b^{(out)}$ los pesos y el bias en la salida, respectivamente.

Algunos casos de datos secuenciales son:

1. Secuencias de imágenes (video)
2. Series de tiempo
3. Lenguaje natural

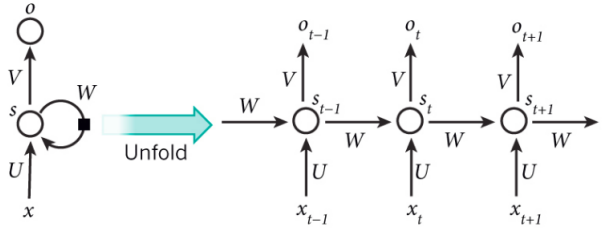
Atención en redes recurrentes

Red recurrente

Una red recurrente es una red que representa los datos secuenciales a partir de recurrencias como:

$$h^{(t)} = g(Wh^{(t-1)} + W'x^{(t)} + b)$$

donde $W \in \mathbb{R}^{d \times n}$ y $b \in \mathbb{R}^n$ son parámetros de la red.

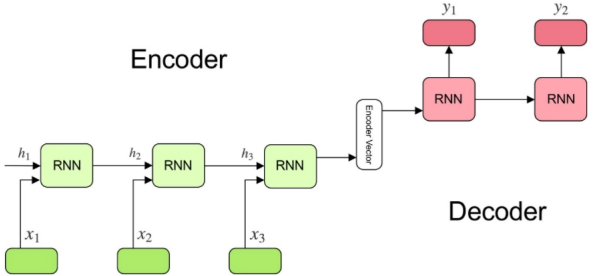


Encoder-decoder

RNN Encoder-Decoder

Una arquitectura encoder-decoder con redes recurrentes es una red neuronal que consta de dos partes:

- **Encoder:** Se encarga de codificar la entrada regresando una codificación de ésta.
- **Decoder:** Decodifica la codificación del encoder para obtener una salida.



Encoder-decoder sequence to sequence model

Atención en RNNs

La codificación del encoder son los vectores $h^{(1)}h^{(2)}\cdots h^{(T)}$, estos se utilizan para crear un vector de codificación que pasa al decoder.
Para poder hacer más eficiente el proceso de codificación se utiliza la **atención**.

Idea de la atención

La atención es un mecanismo que permite, dado un conjunto de elementos de entrada, asignar mayor peso a ciertos elementos que tengan mayo influencia en la salida actual.
Estos pesos $\alpha_{t,k}$ están determinados pro probabilidades (generalmente por medio de la función Softmax).

Atención en RNNs

1. Se aplica el encoder $h^{(1)} h^{(2)} \cdots h^{(T)}$.
2. Por cada salida $s^{(t)}$, en el tiempo t ,se estima:

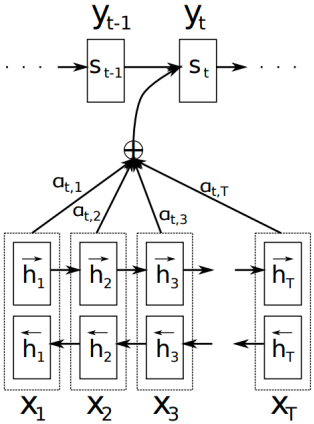
$$sc_{t,k} = e(s^{(t-1)}, h^{(k)})$$

3. Se estiman los pesos de atención:

$$\alpha_{t,k} = Softmax(sc_{t,k})$$

4. se calcula el vector de contexto $c^{(t)}$ cómo:

$$c^{(t)} = \sum_{k=1}^{T} \alpha_{t,k} h^{(k)}$$



Similitud

Para la función $e : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ que determina la similitud entre dos vectores para obtener los pesos de atención, se pueden utilizar diferentes métodos:

- **Producto punto**:

$$sc_{t,k} = s^{(t-1)} \cdot h^{(k)}$$

- **Forma bilineal**: (Luong et al, 2015) Dada una matriz de pesos W:

$$sc_{t,k} = s^{(t-1)} W h^{(k)}$$

- **MLP**: (Bahdanau et al, 2014) Dada una matriz de pesos W y un vector de pesos v:

$$sc_{t,k} = v^T \tanh(W[s^{(t-1)}; h^{(k)} + b])$$

La revolución de la atención

Estos mecanismos de atención presentaron una mejora relevante para modelos tipo **sequence-to-sequence**. Algunas de las ventajas que se presentan son:

- Mejora en el manejo del contexto de las secuencias.
- Capacidad de inferir relaciones en los datos.
- Permite interpretar los pesos de atención como "modelo de traducción".

Sin embargo, la atención representaba uso de capacidades de cómputo y dentro de las RNNs no se podía paralizar este cómputo para eficientar los proceso.

Auto-atención

Motivación de la auto-atención

- Las RNNs tienen la desventaja de que para poder obtener la representación del estado t se necesita obtener antes la del estado $t-1$. **No permiten paralelización**.
- Vaswani et al. (2018) proponen cambiar la forma de representar los datos, **prescindiendo de las recurrencias**.
- La idea fue utilizar **atención para representar** los datos, en lugar de las recurrencias.
- La atención se aplicaría para representar al mismo conjunto de datos, y no entre dos como en las RNNs. A esto se le llamó **auto-atención** (self-attention).

Auto-atención

Pesos de auto-atencioón

Si $x_1, x_2, ...x_n \subseteq \mathbb{R}^d$ es un conjunto de vectores, los pesos de auto-atención se determinan por medio de la función $\alpha : \mathbb{R}^d \times \mathbb{R}^d \to (0, 1)$ como:

$$\alpha(x_i, x_j) = Softmax\Big(\frac{\psi_k(x_i)^T \psi_q(x_j)}{\sqrt{d}}\Big)$$

donde ψ_q y ψ_k son proyecciones de los puntos.

Proyecciones de los datos

Las funciones que denotamos con ψ proyectan los datos a diferentes espacios para poder trabajarlos desde aquí.

Proyección lineal

Las funciones $\psi_q, \psi_k : \mathbb{R}^d \to \mathbb{R}^{d'}$ pueden ser proyecciones lineales definidas como:

$$\psi_q(x_i) = W_q x_i$$
$$\psi_k(x_i) = W_k x_i$$

donde $W_q, W_k \in \mathbb{R}^{d' \times d}$ son matrices de parámetros que se aprenden durante el entrenamiento.

Diremos que los espacios hacia los que se proyectan son el espacio de **queries** y el espacio de **keys**, respectivamente. Se llaman a estos vectores query y key.

Proyecciones y producto punto

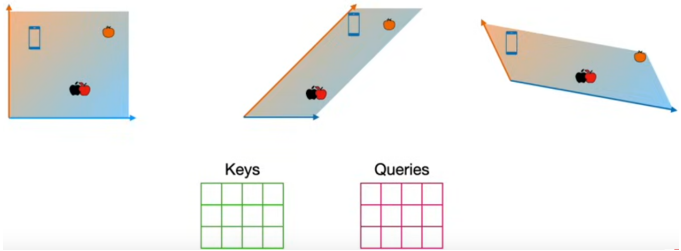
Dada las proyecciones lineales a los espacios de queries y keys podemos observar que:

$$\begin{aligned}
\alpha(x_i, x_j) &= Softmax\Big(\frac{\psi_k(x_i)^T \psi_q(x_j)}{\sqrt{d}}\Big) \\
&= Softmax\Big(\frac{(W_q x_i)^T (W_k x_j)}{\sqrt{d}}\Big) \\
&= Softmax\Big(\frac{x_i^T W_q^T W_k x_j}{\sqrt{d}}\Big) \\
&= Softmax\Big(\frac{x_i^T W x_j}{\sqrt{d}}\Big)
\end{aligned}$$

donde $W = W_q^T W_k$. Se pueden ver las similitudes con la forma bilinear en la atención en RNNs.

Producto punto escalado

El producto punto propuesto por Vaswani et al. (2018) se conoce como **producto punto escalado**, pues escala el resultado por $\sqrt{d}$.
El proyectar los datos originales a diferentes espacios busca que las representaciones sean distintas para la query y la key.



Se puede ver también que se busca aprender ciertas relaciones, pues $x_i^T W x_j = \sum_k \sum_l w_{k,l} x_{i,k} x_{j,l}$

Representaciones con auto-atención

Auto-atención

La auto-atención es una capa para redes neuronales que, dado un conjunto de datos de entrada $x_1 x_2 \cdots x_T$, obtiene una representación de cada dato como:

$$h_i = \sum_{j=1}^{n} \alpha(x_i, x_j) \psi_v(x_j) \tag{1}$$

Donde $\alpha(x_i, x_j)$ es el peso de atención entre x_i y x_j y $\psi_v(x_j)$ es la proyección en el espacio de valores de x_j.

Al igual que en los casos anteriores, la proyección de los valores es lineal:

$$\psi_v(x_j) = W_v x_j$$

Auto-atención

Auto-atención

Si X es la matriz cuyos renglones son los datos, podemos expresar la autoatención como:

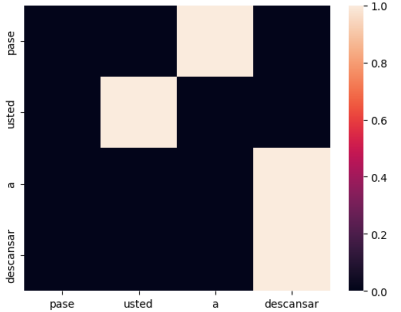
$$Att(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Donde $Q = XW_q^T, K = XW_k^T$ y $V = XW_v^T$.

Se puede notar que:

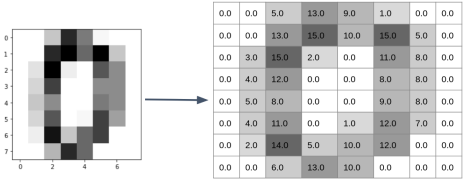
$$Att(Q, K, V)_i = \sum_{j=1}^{n} \alpha(x_i, x_j)\psi_v(x_j)$$

Los pesos de atención $\alpha(x_i, x_j)$ pueden interpretarse como la probabilidad de la relación entre ambos elementos x_i y x_j.

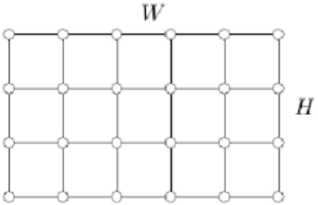


Estructura subyacente de los datos

-El aprendizaje profundo se basa fuertemente en **aprendizaje representacional**. Busca obtener representaciones h que puedan llevar a solucionar el problema.
-Cuando se trata de datos con estructuras complejas, las representaciones aprendidas deben aprovechar información sobre la **esructura de los datos**.
-Por ejemplo, las redes convolucionales se fijan en píxeles vecinos:



(a) Representación matricial de una imagen

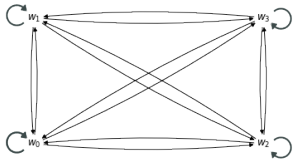


(b) Estructura gráfica de cuadrícula

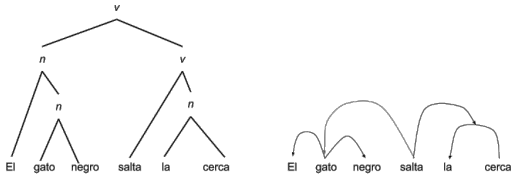
Auto-atención y estructura gráfica

Suposición de la auto-atención

El modelo de auto-atención asume una gráfica $G = (V, E)$ completamente conectada con un conjunto de vértices asociados a los vectores de entrada $x_1, ..., x_n \in \mathbb{R}^d$.
Podemos suponer una matriz pesada, donde los pesos están determinados por la atención $\alpha(x_i, x_j)$

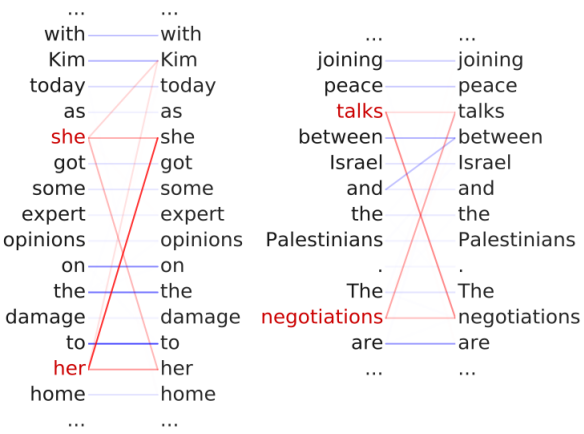


(a) Gráfica completamente conectada



(b) Estructuras gráficas del lenguaje

Relaciones en auto-atención



Se ha visto que los modelos de auto-atención pueden capturar diferentes relaciones entre los tókens de entrada (Clark et al., 2019). Algunas de las relaciones encontradas son:

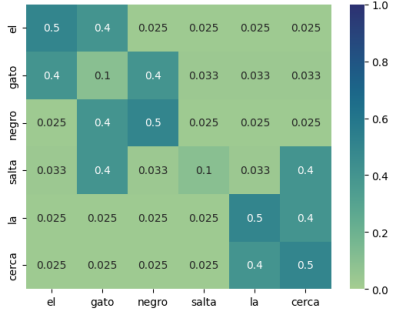
- Objetos directos
- Modificadores de sustantivos
- Pronombres posesivos
- Auxiliares de verbos
- Preposiciones
- Correferencias y anáforas

Pesos de atención

Matriz de adyacencia y pesos de atención

Si asumimos que la auto-atención asume una estructura gráfica, podemos definir una matriz de adyacencia A definida como:

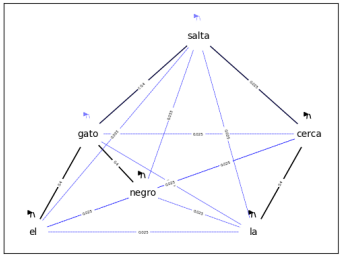
$$A_{i,j} = \alpha(x_i, x_j)$$



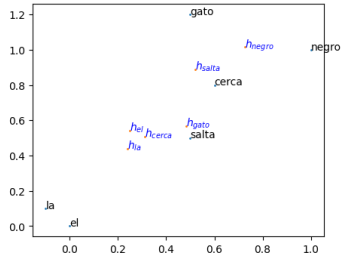
Auto-atención, gráficas y representación

La auto-atención lleva las relaciones de la gráfica pesada a una representación **convexa** sobre el espacio de valores.
Los nuevos vectores siempre quedarán dentro del conjunto convexo formado por las proyecciones de los tókens en los values.

$$h_{negro} = 0.025 \cdot v_{el} + 0.4 \cdot v_{gato} + 0.5 \cdot v_{negro} + 0.025 \cdot v_{salta} + 0.025 \cdot v_{la} + 0.025 \cdot v_{cuerda}$$



(a) Gráfica con pesos de atención



(b) Representación obtenida

Redes gráficas y auto-atención

Auto-atención gráfica

Sea $x_1, x_2, \cdots x_n \subseteq \mathbb{R}^d$ un conjunto de puntos asociados a los nodos de una gráfica $G = (V, E)$. Si $\mathcal{N}_i$ son los vecinos en G de x_i, podemos definir la capa de auto-atención como:

$$h_i = \sum_{j \in \mathcal{N}_i} \alpha(x_i, x_j) \psi_v(x_j) \tag{2}$$

Nota

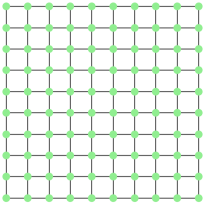
Si G es una **gráfica completamente conectada**, entonces:

$$\begin{aligned} h_i &= \sum_{j \in \mathcal{N}_i} \alpha(x_i, x_j) \psi_v(x_j) \\ &= \sum_{j} \alpha(x_i, x_j) \psi_v(x_j) \end{aligned}$$

Estructura de cuadrícula

Cuadrícula

Una cuadrícula es una gráfica donde los nodos se ordenan en coordenadas $(i \times j)$ y dos nodos están conectados entre sí si están a distancia 1.



Si asociamos índices $x_{i,j}$ al punto en la coordenada (i, j), observamos que tiene como vecinos, entonces:

$$\mathcal{N}_{i,j} = \{x_{i+h,j+w} : h, w \in \{-1, 0, 1\}\}$$

De tal forma que la capa de atención se podría expresar como:

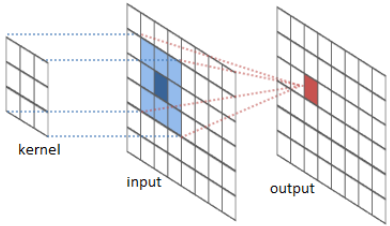
$$\begin{aligned} h_{i,j} &= \sum_{k \in \mathcal{N}_{i,j}} \alpha(x_{i,k}, x_k)\psi_v(x_k) \\ &= \sum_h \sum_w \alpha(x_{i,j}, x_{i+h,j+w})\psi_v(x_{i+h,j+w}) \end{aligned}$$

Redes convolucionales y auto-atención

Redes convolucionales

Bronstein et al. (2021) sugieren que las redes convolucionales son redes de auto-atención sobre una gráfica de tipo cuadrícula. Tal que la proyección ψ_v es la identidad y los pesos de atención son valores no acotados:

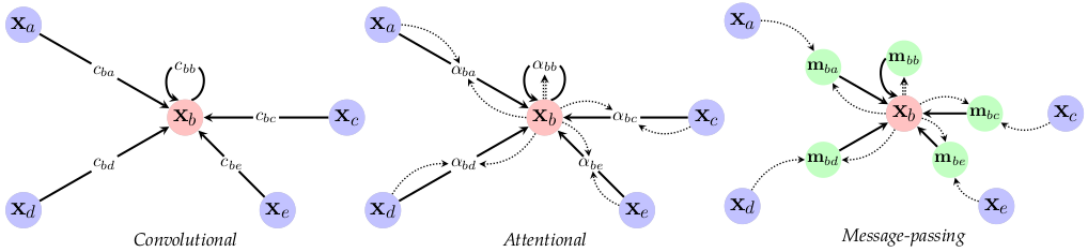
$$h_{i,j} = \sum_h \sum_w \alpha_{u,v} x_{i+h,j+w} + b$$



Otras estructuras gráficas

Bajo los supuestos establecidos, se puede asumir que las redes que hacen uso de capas de **auto-atención** son **redes gráficas** y en particular reds de tipo Message-Passing.

- La auto-atención asume una gráfica completamente conectada.
- La atención (encoder-decoder) asume una gráfica bipartita.



Encoder-Decoder

El problema de la posición

Problemática

Al tratar con los tókens de una cadena del lenguaje natural, la auto-atención aprenderá relaciones entre estos tókens.
Sin embargo, no tiene información de la **secuencia** que tienen los tókens. El **lenguaje es lineal** en principio.

Por ejemplo, la siguiente oración tokenizada:

el	niñ#	#o	jug#	#aba	con	lo#	#s	animal#	#es
$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

será considerada como **bolsa de palabras**.

Codificación posicional

Codificiación posicional

Un vector de codificación posicional es un vector en $\mathbb{R}^d$ que codifica la posición t de un tóken en la cadena de entrada a partir de funciones senos y cosenos:

$$pe_{2t} = \sin\left(\frac{t}{warm^{2t/d}}\right)$$

$$pe_{2t+1} = \cos\left(\frac{t}{warm^{2t/d}}\right)$$

Donde *warm* o warmup es un hiperparámetro.

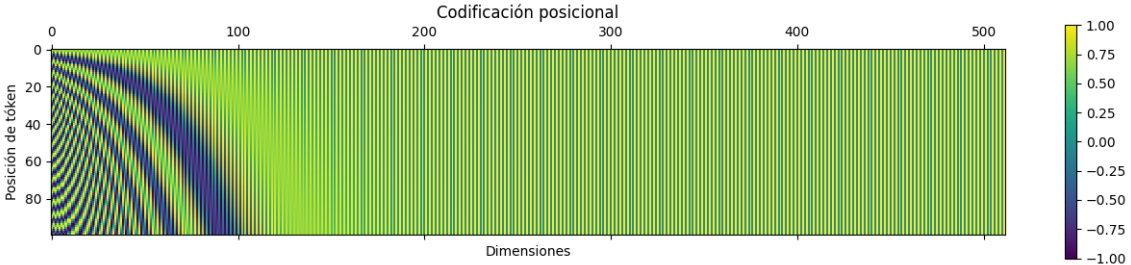
Generalmente se asume que:

$$warm = 10000$$

Entrada de atención

En el encoder, se toma como entrada los vectores de **embeddings** escalados y se suma el vector **posicional**:

$$x_i = \sqrt{d}e_i + pe_i$$



Codificación posicional absoluta

A la codificación posiciona que hemos revisado (Vaswani et al., 2017) se le conoce como **codificación posicional absoluta**, y establece las proyecciones de los datos como:

$$q_i = W_q(e_i + pe_i)$$
$$k_i = W_k(e_i + pe_i)$$
$$v_i = W_v(e_i + pe_i)$$

La información posicional se basa en la posición **absoluta** del tóken dentro de su contexto.

Codificación posicional relativa

Distancia relativa

Sean w_i y w_j dos tókens con posiciones absolutas i y j, respectivamente, la distancia relativa (o clip) se estima como:

$$clip(j-i,k) = \max\{-k, \min\{j-i,k\}\}$$

donde k es la posición relativa máxima.

Adyacencia posicional

Sean $k_1,..,k_n$ y $v_1,...,v_n$ las proyecciiones en keys y values, respectivamente. Definimos las matrices de adyacencia como:

$$a^k_{i,j} = w^k_{clip(j-i,k)}$$
$$a^v_{i,j} = w^v_{clip(j-i,k)}$$

donde $w^k_{-k},...,w^k_0,...,w^k_k$ y $w^v_{-k},...,w^v_0,...,w^v_k$ son parámetros de la red.

Codificación posicional relativa

Codificaicón posicional relativa

La codificación posicional relativa (Shaw et al., 2021) utiliza la adyacencia posicional para estimar los pesos de auto-atención como:

$$\alpha(x_i, x_j) = Softmax\Big(\frac{(W_q x_i)^T (W_k x_j + a^k_{i,j})}{\sqrt{d}}\Big)$$

Asimismo, las representaciones en la auto-atención se obtienen como:

$$h_i = \sum_j \alpha(x_i, x_j)\big(W_v x_j + a^v_{i,j}\big)$$

Embeddings rotacionales

Otra alternativa para la codificación posicional son los embeddings rotacionales (Su et al., 2024) donde:

$$q_i = R^d_{\Theta,i} W_q x_i$$
$$k_i = R^d_{\Theta,i} W_q x_i$$

Los embeddings rotacionales capturan información de posición de i con respecto a j como efecto del producto punto:

$$\begin{aligned} q_i^T \cdot k_j &= \left(R^d_{\Theta,i} W^{(q)} x_i\right)^T \left(R^d_{\Theta,j} W^{(k)} x_j\right) \\ &= x_i^T W^{(q)} \mathbf{R}^{\mathbf{d}}_{\mathbf{,j-i}} W^{(k)} x_j \end{aligned}$$

Con la matriz de rotación:

$$R^d_{\Theta,i} = \begin{pmatrix} \cos i\theta_1 & -\sin i\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin i\theta_1 & \cos i\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos i\theta_2 & -\sin i\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin i\theta_2 & \cos i\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos i\theta_{d/2} & -\sin i\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & -\sin i\theta_{d/2} & \cos i\theta_{d/2} \end{pmatrix}$$

Necesidad de normalización

Problema de cambio interno de covarianza

El problema de cambio interno de covarianza el cambio en las activaciones de la red debido al cambio de los parámetros de la red neuronal durante el entrenamiento

Solución

Estabilizar las entradas en cada capa puede ayudar a que los valores de la activación se alejen de los límites de esta, permitiendo un mejor entrenamiento.

La estabilización de las entradas requiere de su normalización con respecto a los lotes de entrada.

Normalización por lotes

Normalización por lotes

Dado un dato $x \in \mathbb{R}^d$ de un lote, este se normaliza como:

$$\hat{x} = a \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + b$$

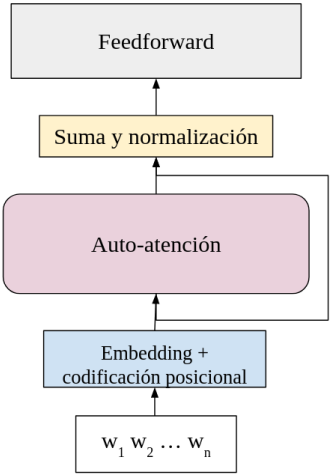
donde a y b son parámetros y ϵ evita la división entre 0.

La media y la varianza se calcula de la manera usual:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

$$\sigma^2 = \frac{1}{N-1} \sum_{i} (x_i - \mu)^2$$

Suma y normalización



En los módulos tanto de encoder como decoder, la normalización se sigue:

- Se aplica una **normalización** a los vectores (de la capa previa).
- Se aplica la capa actual.
- Se hace una **conexión redisual** con los vectores de la capa previa.

De tal forma que se tiene en cada capa:

$$h = x + capa(norm(x))$$

Atención multi-cabeza

Cabeza de (auto-)atención

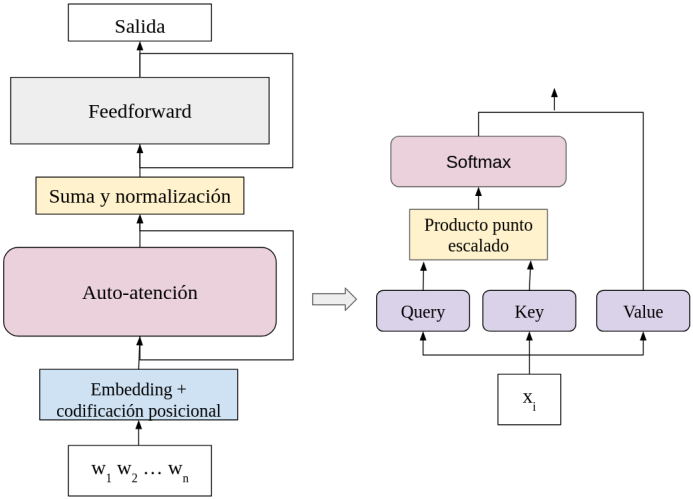
Entenderemos por una cabeza de auto-atención a la capa con la cual se obtiene la matriz de representaciones dada como:

$$h_{1:T} = Att(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3}$$

En un modelo de auto-atención de una cabeza contamos con:

- Embeddings y codificación posicional.
- Cabeza de auto-atención.
- Suma y normalización entre cada capa.

Modelo de atención con una cabeza



Multi-cabeza

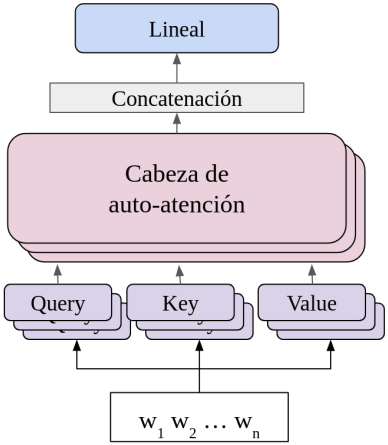
Atención multi-cabeza

La atención multi-cabeza asume el uso de N cabezas de atención, cada una de la forma:

$$head_i = Att(Q_i, K_i, V_i) = Softmax\Big(\frac{Q_i K_i^T}{\sqrt{d_i}}\Big) V_i$$

Con sus propios pesos de query, key y value. La representación final será:

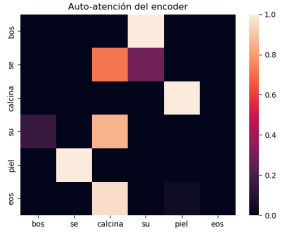
$$h = [head_1 || head_2 || \cdots || head_N] W + (b)$$



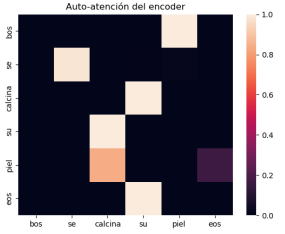
Representración en multi-cabezas

La auto-atención multi-cabeza es combinación afín de las cabezas de auto-atención. Supongamos que la matriz W tienen N secciones $W_i \in \mathbb{R}^{d \times d}$ tal que $W = [W_1||W_2||\cdots||W_N]$, entonces, la representación de la fórmula anterior puede expresarse como:

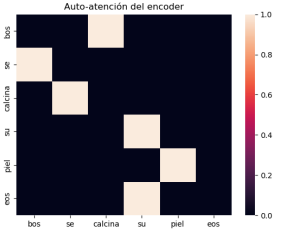
$$h = head_1 W_1 + head_2 W_2 + \cdots + head_N W_N + (b)$$



(a) Primera cabeza



(b) Segunda cabeza



(c) Tercera cabeza

Otras propuestas de muti-cabeza

Atención colaborativa

La atención colaborativa (Cordonier et al, 2020) define una sola proyección query y key, y N proyecciones value V_i. Cada cabeza se calcula como:

$$head_i = Att(QM_r, K, V_i) = Softmax\Big(\frac{QM_rK^T}{\sqrt{d}}\Big)V_i$$

Donde M_r es una matriz diagonal tal que:

$$(QM_rK^T)_{i,j} = \sum_l m_{r,l}\phi_q(x_i)_l\phi_k(x_j)_l$$

Enmascaramiento

Problema

El objetivo del decoder es obtener la probabilidad de un tóken w_i, dado los elementos pasados $w_1, \cdots, w_{i-1}$, por lo que realizar un mecanismo de atención en el que se observan **tókens futuros** introduce un sesgo.

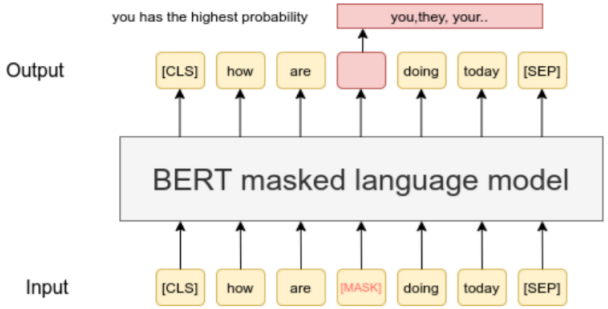
Enmascaramiento

El enmascaramiento reemplaza un tóken w_i por una etiqueta `MASK` para que el modelo no tenga información de esta etiqueta. Este proceso oculta un tóken para que no sea accesible a la auto-atención.

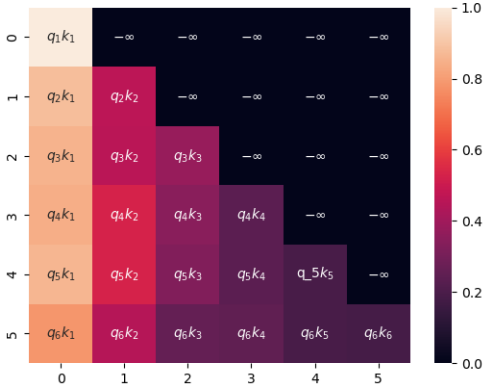
Modelos enmascarados

Enmascaramiento de tókens

Modelos como BERT (Devlin et al., 2019) enmascaran tókens de una cadena de entrada para buscar predecir la palabra en su contexto, usando tanto los elementos previos como los subsecuentes.



Enmascaramiento en decoder



Enmascaramiento subsecuente

Para predecir los tókens subsecuentes (tarea del decoder) se enmascaran todas las relaciones con los tókens subsecuentes. De tal forma que las relaciones del tóken w_i estarán dadas por:

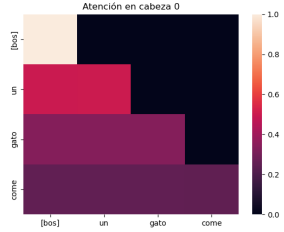
$$\mathcal{N}_i = \{j : 1 \leq j \leq i\}$$

Modelos con enmascaramiento

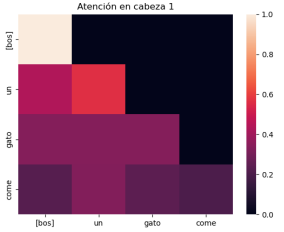
Auto-atención enmascarada

La auto-atención con enmascaramiento subsecuente estima las representaciones de las entradas como:

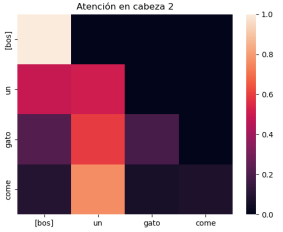
$$h_i = \sum_{j \leq i} \alpha(x_i, x_j)\psi_v(x_j)$$



(a) Primera cabeza



(b) Segunda cabeza



(c) Tercera cabeza

Atención en gráficas

El proceso de enmascaramiento permite evitar las relaciones con los elementos subsecuentes. Sin embargo, no establece relaciones complejas entre los elementos de la entrada.

Atención sobre gráficas

En general, un método de auto-atención en gráficas en donde se toma en cuenta sólo los elementos relacionados en una gráfica no necesariamente conectada por completo. Esto es:

$$h_i = \sum_{j \in \mathcal{N}_i} \alpha_(x_i, x_j) \psi_v(x_j)$$

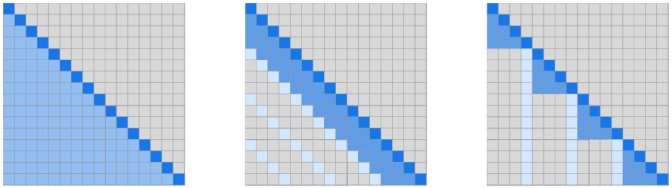
- El problema de este tipo de atención es el que requiere de información explícita de la gráfica.
- Además, eleva el costo computacional al ser una red neuronal gráfica.

Atención dispersa

Atención dispersa

La atención dispersa (Child et al., 2019) propone generar matrices dispersa de atención, resaltando relaciones no completamente conectadas en los elementos de entrada, pero que no requieran de una estructura gráfica implícita.
Por ejemplo **stride attention** que asume vecindades de la forma:

$$\mathcal{N}_i = \{j : max(0, i - k) \leq j \leq i\}$$



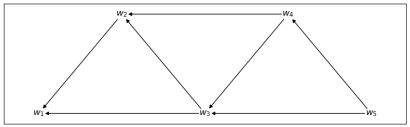
(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Stride Attention

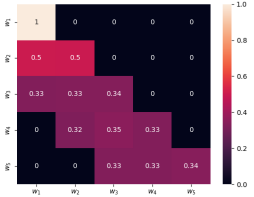
Stride Attention

El mecanismo de stride attention estima la atención con base en un hiperparámetro k tal que:

$$h_i = \sum_{max(0,i-k) \leq j \leq i} \alpha(x_i, x_j)\psi_v(x_j)$$



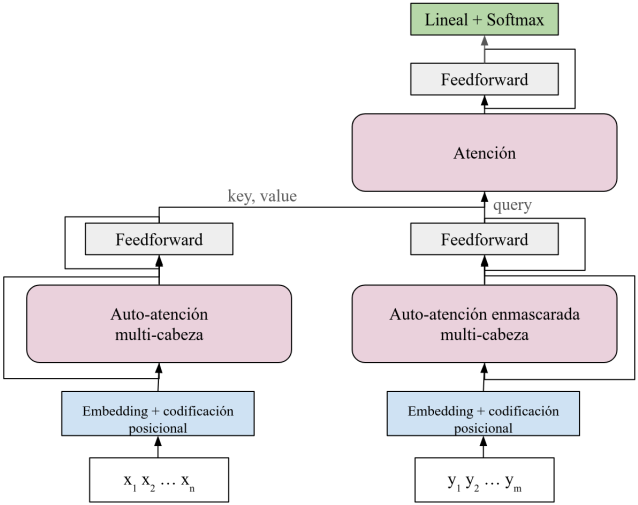
(a) Estructura gráfica



(b) Matriz dispersa

Transformador

Estructura del Transformador

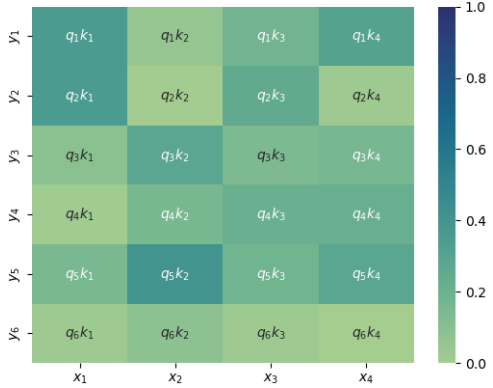


Atención Encdoer-Decoder

Atención (encoder-decoder)

Sean $x_1 x_2 \cdots x_n$ las representaciones del encoder y $y_1 y_2 \cdots y_m$ las del decoder, entonces, la atención entre el encoder y el decoder se estima como:

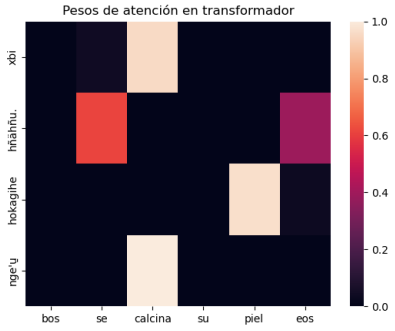
$$h_j = \sum_i \alpha(y_j, x_i)\psi_v(x_i)$$



Atención Encoder-Decoder

Los pesos de atención se estiman a partir de la proyección de los datos del encoder en el espacio de keys y values, mientras que los datos de salida se proyectan al espacio de queries.

$$\alpha(y_i, x_j) = Softmax_x\left(\frac{\psi_q(y_i)^T \psi_k(x_j)}{\sqrt{d}}\right)$$



Salida del Trasnformador

Salida

Después de pasar la atención entre encoder y decoder, las representaciones de las salidas se pasan por una red Feedforward, obteniendo las representaciones finales $h_{1:t}$. La salida aplica una capa lineal y una activación Softmax para obtener la predicción:

$$\hat{p}(y_{t+1}|y_{1:t}, x_{1:n}) = Softmax(Wh_{1:t} + b)$$

Función objetivo

La función objetivo suele estar dada por la entropía cruzada:

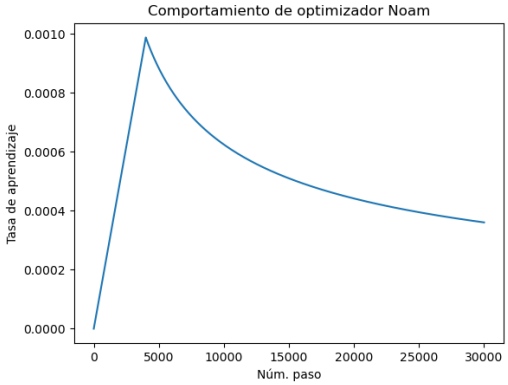
$$\mathcal{J} = -\sum_{x_{1:n}} \sum_{t} \delta_{y,t} \ln \hat{p}(y_{t+1}|y_{1:t}, x_{1:n})$$

Optimización

Optimizador Noam

El optimizador Noam (Vaswani et al., 2017) es un optimizador basado en Adam en donde la tasa de aprendizaje se adapta de acuerdo a la regla:

$$\eta = \frac{1}{\sqrt{d}} \min\{\frac{1}{\sqrt{t}}, \frac{t}{\sqrt{\omega^3}}\}$$



Referencias

- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
- Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. Advances in neural information processing systems, 31.
- Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478.
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... & Fiedel, N. (2023). Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240), 1-113.
- Cordonnier, J. B., Loukas, A., & Jaggi, M. (2020). Multi-head attention: Collaborate instead of concatenate. arXiv preprint arXiv:2006.16362.
- Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does bert look at? an analysis of bert's attention. arXiv preprint arXiv:1906.04341.
- Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509.
- Dar, G., Geva, M., Gupta, A., & Berant, J. (2022). Analyzing transformers in embedding space. arXiv preprint arXiv:2209.02535.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.
- Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456).
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- Knyazev, B., Taylor, G. W., & Amer, M. (2019). Understanding attention and generalization in graph neural networks. Advances in neural information processing systems, 32.
- Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., & Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568, 127063.
- Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. arXiv preprint arXiv:1803.02155.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. arXiv preprint arXiv:1710.10903.

¡Gracias!

vmijangosc@ciencias.unam.mx