Normalización de transformers

GLyC

September 23, 2025

Vamos a definir dos nociones de normalización de los transformers. Ambas van a hablar del valor del último vector antes de salir del ciclo. Lo que en el paper llamarían h_n^L. Además la matriz de output será $\sim$ la identidad.

Normalización 0-1

Definición

Vamos a decir que un transformer T_N es la 0-1 normalización de un transformer T si el h_n^L de T_N cumple que:

$$(h_n^L)_i = \begin{cases} T_{NO} & \text{si } i = 1 \\ T_{SI} & \text{si } i = 2 \\ 0 & \text{cc} \end{cases}$$

donde T_{NO} y T_{SI} corresponden a los valores que calcula la función $OUTPUT$ antes de hacer el softmax, es decir

$$T_{NO} := (\Theta_{OUTPUT}(h_n^L))_1$$
$$T_{SI} := (\Theta_{OUTPUT}(h_n^L))_2$$

Implementación

Vamos a tomar un transformer T y obtener su normalizado 0-1. Para ello lo que vamos a hacer es agregar algunas capas más al final de T para conseguir aplicarle una transformación lineal a h_n^L.

Dado que $\Theta_{OUTPUT} \in \mathbb{R}^{2\times d}$ basta con extender Θ_{OUTPUT} con 0 hasta tener $\Theta'_{OUTPUT} \in \mathbb{R}^{d\times d}$

$$\Theta_{OUTPUT} h_n^L = \begin{pmatrix} T_{NO} \\ T_{SI} \end{pmatrix} \longrightarrow \begin{pmatrix} \Theta_{OUTPUT} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} h_n^L = \begin{pmatrix} T_{NO} \\ T_{SI} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Luego la matriz Θ_{OUTPUT} de este nuevo transformer sería:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \end{pmatrix}$$

Dado que

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} \Theta_{OUTPUT}h & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} h =$$

$$= \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} T_{NO} \\ T_{SI} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} T_{NO} \\ T_{SI} \end{pmatrix}$$

Por lo tanto preserva el comportamiento.

Falta ver es cómo aplicarle una transformación lineal a un vector y guardarla al final. Más sobre eso en Requisitos.

Normalización ∞

Definición

Vamos a decir que un transformer T_N es la normalización de un transformer T si el h_n^L de T_N cumple que:

repensar está definición: no sería mejor que sea algo tipo $\infty, -\infty$ si responde no y $-\infty, \infty$ si responde sí? El comportamiento como estamos tomando argmax sería el mismo

$$(h_n^L)_i = \begin{cases} 1 & \text{si } i = 1 \\ x & \text{si } i = 2 \\ 0 & \text{cc} \end{cases} \qquad \text{donde } x = \begin{cases} 0 & \text{si T responde no} \\ \infty & \text{si T responde sí} \end{cases}$$

Implementación

La matriz de $OUTPUT$ del transformer $\infty-$normalizado va a ser la misma que para el 0-1 normalizado. Veamos cómo conseguir ahora que el último vector sea lo deseado.

Sin pérdida de generalidad podemos ∞ normalizar un transformer 0-1 normalizado, por ende podemos suponer que $h_n^L = (T_{NO}, T_{SI}, 0, \dots, 0)$

Primero vamos a obtener un vector de la forma $(0, x, 0, \dots, 0)$ donde valdrá que $x = 0$ si T respondía NO y $x > 0$ si T respondía SI. Notar que si T responde SI entonces $T_{SI} > T_{NO}$ y que si T responde NO entonces $T_{NO} > T_{SI}$.

Al pasar por una capa de FF con $W_2 = Id$, $b1 = 0$, $b2 = 0$ y W_1 con todos ceros salvo los dos elementos de la última fila los cuales valdrían -1 y 1. Obtenemos algo cercano a lo deseado:

$$Id\,\mathrm{relu}\left(\begin{pmatrix} 0 & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & 0 \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} T_{NO} \\ T_{SI} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + 0\right) + 0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathrm{relu}(T_{SI} - T_{NO}) \end{pmatrix}$$

Dado que la FF se suma a lo que ya teníamos nuestro n-ésimo vector ahora es de la forma: $(T_{NO}, T_{SI}, 0, \dots, 0, \mathrm{relu}(T_{SI} - T_{NO}))$. Notar que:

$$\mathrm{relu}(T_{SI} - T_{NO}) \begin{cases} = 0 \text{ si } T \text{ responde NO} \\ > 0 \text{ si } T \text{ responde SI} \end{cases}$$

Dado que podemos aplicarle transformaciones lineales a un vector podemos colocar en 0 las dos primeras coordenadas basta aplicarle la identidad con 0 en los primeros dos elementos de la diagonal. Luego ya tendríamos un vector de la pinta $(0, \dots, 0, \mathrm{relu}(T_{SI} - T_{NO}))$. Con una permutación podemos transformarlo en $(0, \mathrm{relu}(T_{SI} - T_{NO}), 0, \dots, 0)$.

Por la representación finita ocurre que dado dado un x no negativo vale:

$$x * \infty = \begin{cases} 0 \text{ si } x = 0 \\ y \text{ para algún } y \geq 1 \text{ si } x \geq 1 \text{ pues } x \geq \frac{1}{\infty} \end{cases}$$

Luego

$$(x * \infty) * \infty = \begin{cases} 0 \text{ si } x = 0 \\ \infty \text{ si } x \geq 1 \end{cases}$$

Por lo tanto aplicarle la transformación lineal que multiplica por ∞ dos veces al vector que tenemos nos da el resultado esperado pero con 0 en la primer coordenada. Lo cual se puede solucionar con una capa de FF donde $W_2 = W_1 = Id$, $b_1 = 0$ y $b_2 = (1, 0, \dots, 0)$.

Tomando como matriz de output la identidad que proyecta las primeras dos coordenadas es evidente ver que el comportamiento de este nuevo transformer es igual al del original:

- Si T respondía NO entonces la primer coordenada será 1 y la segunda 0, por lo tanto el argmax hace que el transformer ∞-normalizado devuelva NO (pues está asociado a la primer coordenada, la cual es mayor que la segunda).
- Si T respondía NO entonces la primer coordenada será 1 y la segunda ∞, por lo tanto el argmax hace que el transformer ∞-normalizado devuelva SI (pues está asociado a la segunda coordenada la cual es mayor que la primera).

Requisitos

Embedding con flags

flagear vectores con el positional embedding

Hacer una capa 0

0

ATTN y FF controlando que coordenadas afectamos

preservación

Transformaciones lineales

Vamos a ver cómo aplicarle una transformación lineal a uno de nuestros vectores y guardar su resultado en otro vector más a la izquierda, es decir: $h_j^{l+c} = Mh_i^l$ para un par $i < j$ elegidos previamente. Notar que el procedimiento que presentamos acá rompe todos los vectores salvo el h_j es decir que no vamos a ser capaces de predecir qué valores tendrán los demás.

Primero vamos hacer que $h_j = 0$ en las coordenadas que nos interesan. Luego con una capa de ATTN le vamos a sumar Mh_i.

Por las secciones anteriores podemos asumir que h_i es el único vector que sus últimas coordenadas son $(1, 0)$ y que todos los demás terminan en $(0, 1)$. Estos números habrían sido puestos por el encoding y preservados en las capas anteriores.

Esta información nos va a permitir construir W_K y W_Q tales que:

$$\langle q_j, k_{i'} \rangle = \begin{cases} -\infty & \text{si } i' \neq i \\ 0 & \text{si } i' = i \end{cases}$$

Notar que las W_Q y W_K que buscamos son las siguientes

$$W_Q = \begin{pmatrix} 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & -\infty \end{pmatrix} \qquad W_K = \begin{pmatrix} 0 & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

dado que

$$q_j = W_Q\, h_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\infty \end{pmatrix} \qquad k_{i'} = W_K\, h_{i'} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ (h_{i'})_d \end{pmatrix}$$

pues ahora $\langle q_j, k_{i'} \rangle = -\infty * (h_{i'})_d$

Luego usando $W_O = M$ y $W_V = id$ obtendríamos lo esperado.

Dado que $e^{-\infty} = 0$ ocurre que $softmax(-\infty, \ldots, -\infty, 0) = (0, \ldots, 0, 1)$. Lo cual genera que $(s_j)_{i'} = \mathbb{I}(i' = i)$ y por ende:

$$W_O \sum_{i'=0}^{n} (s_j)_{i'} v_{i'} = W_O \sum_{i'=0}^{j} (s_j)_{i'} v_{i'} = W_O \sum_{i'=0}^{j} (s_j)_{i'} h_{i'} = W_O \sum_{i'=0}^{j} \mathbb{I}(i' = i) h_{i'} = W_O \ h_i$$