La atención es todo lo que necesitas

Ashish Vaswani *
Google Brain
avaswani@google.com

Noam Shazeer *
Google Brain
noam@google.com

Niki Parmar *
Google Research
nikip@google.com

Jakob Uszkoreit *
Google Research
usz@google.com

Llion Jones *
Google Research
llion@google.com

Aidan N. Gomez * †
Universidad de Toronto
aidan@cs.toronto.edu

Łukasz Kaiser *
Google Brain
lukaszkaiser@google.com

Illia Polosukhin * ‡
illia.polosukhin@gmail.com

Resumen

Los modelos de transducción de secuencia dominantes se basan en redes neuronales recurrentes o convolucionales complejas que incluyen un codificador y un decodificador. Los modelos con mejor rendimiento también conectan el codificador y el decodificador a través de un mecanismo de atención. Proponemos una nueva arquitectura de red simple, el Transformer, basada únicamente en mecanismos de atención, prescindiendo por completo de la recurrencia y las convoluciones. Los experimentos en dos tareas de traducción automática muestran que estos modelos son superiores en calidad, a la vez que son más paralelizables y requieren mucho menos tiempo de entrenamiento. Nuestro modelo alcanza 28,4 BLEU en la tarea de traducción de inglés a alemán WMT 2014, mejorando los mejores resultados existentes, incluidos los conjuntos, en más de 2 BLEU. En la tarea de traducción de inglés a francés WMT 2014, nuestro modelo establece una nueva puntuación BLEU de última generación de un solo modelo de 41,8 después de entrenar durante 3,5 días en ocho GPU, una pequeña fracción de los costes de entrenamiento de los mejores modelos de la literatura. Demostramos que el Transformer se generaliza bien a otras tareas aplicándolo con éxito al análisis sintáctico constituyente inglés tanto con datos de entrenamiento grandes como limitados.

* Contribución equitativa. El orden de la lista es aleatorio. Jakob propuso reemplazar las RNN con autoatención y comenzó el esfuerzo por evaluar esta idea. Ashish, con Illia, diseñó e implementó los primeros modelos Transformer y ha participado de manera crucial en todos los aspectos de este trabajo. Noam propuso la atención de producto escalar escalada, la atención de múltiples cabezas y la representación de posición sin parámetros y se convirtió en la otra persona involucrada en casi todos los detalles. Niki diseñó, implementó, ajustó y evaluó innumerables variantes de modelos en nuestra base de código original y tensor2tensor. Llion también experimentó con variantes de modelos novedosas, fue responsable de nuestra base de código inicial y de la inferencia y visualizaciones eficientes. Lukasz y Aidan pasaron incontables días diseñando varias partes e implementando tensor2tensor, reemplazando nuestra base de código anterior, mejorando enormemente los resultados y acelerando masivamente nuestra investigación.

† Trabajo realizado mientras estaba en Google Brain.
‡ Trabajo realizado mientras estaba en Google Research.

31ª Conferencia sobre Sistemas de Procesamiento de Información Neuronal (NIPS 2017), Long Beach, CA, EE. UU.

1 Introducción

Redes neuronales recurrentes, memoria a corto plazo [13] y redes neuronales recurrentes cerradas [7] en particular, se han establecido firmemente como enfoques de vanguardia en problemas de modelado y transducción de secuencias, como el modelado de lenguajes y la traducción automática [35 2 5]. Numerosos Desde entonces, se han seguido realizando esfuerzos para superar los límites de los modelos de lenguaje recurrentes y las arquitecturas de codificador-decodificador [38, 24, 15].

Los modelos recurrentes suelen factorizar la computación a lo largo de las posiciones de los símbolos de las secuencias de entrada y salida. Al alinear las posiciones con los pasos en el tiempo de computación, generan una secuencia de estados ocultos h t , como una función del estado oculto anterior h t y la entrada para la posición t . Esto inherentemente La naturaleza secuencial impide la paralelización dentro de los ejemplos de entrenamiento, lo que se vuelve crítico en longitudes de secuencia más largas, ya que las restricciones de memoria limitan la agrupación en lotes entre ejemplos. El trabajo reciente ha logrado mejoras significativas en la eficiencia computacional a través de trucos de factorización [21] y condicional computación [32], al tiempo que mejora el rendimiento del modelo en el caso de este último. Lo fundamental La restricción de la computación secuencial, sin embargo, permanece.

Los mecanismos de atención se han convertido en una parte integral del modelado de secuencias convincente y los modelos de transducción en varias tareas, lo que permite modelar las dependencias sin tener en cuenta su distancia en las secuencias de entrada o salida [2 19]. En todos, salvo en algunos casos [27], sin embargo, tales mecanismos de atención se utilizan junto con una red recurrente.

En este trabajo proponemos el Transformer, una arquitectura de modelo que evita la recurrencia y, en cambio, se basa completamente en un mecanismo de atención para establecer dependencias globales entre la entrada y la salida. El Transformer permite una paralelización significativamente mayor y puede alcanzar un nuevo estado del arte en la calidad de la traducción después de ser entrenado durante tan solo doce horas en ocho GPU P100.

2 Fondo

El objetivo de reducir la computación secuencial también forma la base de la GPU neuronal extendida [16], ByteNet [18] y ConvS2S [9], todos los cuales utilizan redes neuronales convolucionales como base bloque, computando representaciones ocultas en paralelo para todas las posiciones de entrada y salida. En estos modelos, el número de operaciones necesarias para relacionar señales de dos posiciones de entrada o salida arbitrarias crece en la distancia entre posiciones, linealmente para ConvS2S y logarítmicamente para ByteNet. Esto hace que sea más difícil aprender dependencias entre posiciones distantes [12]. En el Transformer esto es reducido a un número constante de operaciones, aunque a costa de una resolución efectiva reducida debido al promedio de posiciones ponderadas por atención, un efecto que contrarrestamos con la atención multi-cabeza como se describe en la sección 3.2.

La autoatención, a veces llamada intra-atención, es un mecanismo de atención que relaciona diferentes posiciones de una sola secuencia para calcular una representación de la secuencia. La autoatención se ha utilizado con éxito en una variedad de tareas, incluida la comprensión de lectura, el resumen abstracto, la implicación textual y el aprendizaje de representaciones de oraciones independientes de la tarea [4, 27, 28, 22].

Las redes de memoria de extremo a extremo se basan en un mecanismo de atención recurrente en lugar de recurrencia alineada con la secuencia y se ha demostrado que funcionan bien en tareas de preguntas y respuestas en lenguaje simple y modelado de lenguaje [34].

Sin embargo, hasta donde sabemos, el Transformer es el primer modelo de transducción que se basa completamente en la autoatención para calcular las representaciones de su entrada y salida sin utilizar RNN o convolución alineadas con la secuencia. En las siguientes secciones, describiremos el Transformer, motivaremos la autoatención y discutiremos sus ventajas sobre modelos como [17, 18] y [9].

3 Arquitectura del modelo

La mayoría de los modelos de transducción de secuencia neuronal competitivos tienen una estructura de codificador-decodificador [5 2 35]. Aquí, el codificador asigna una secuencia de entrada de representaciones de símbolos $(x_1, ..., x_n)$ a una secuencia de representaciones continuas $z = (z_1, ..., z_n)$. Dado z, el decodificador luego genera una salida secuencia $(y_1, ..., y_m)$ de símbolos un elemento a la vez. En cada paso, el modelo es auto-regresivo [10], consumiendo los símbolos generados previamente como entrada adicional al generar el siguiente.

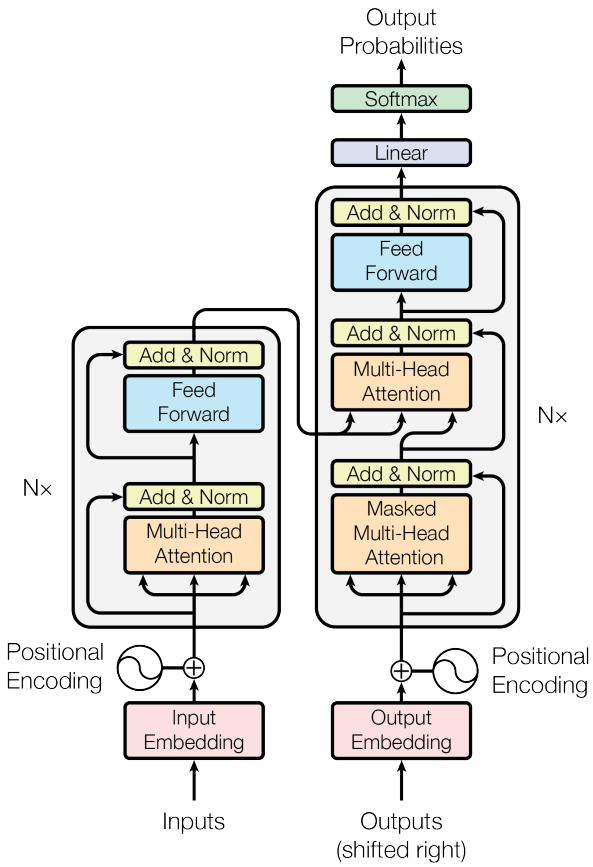

Figura 1: El Transformador - arquitectura del modelo.

El Transformador sigue esta arquitectura general utilizando auto-atención apilada y capas totalmente conectadas punto a punto para el codificador y el decodificador, que se muestran en las mitades izquierda y derecha de la Figura 1, respectivamente.

3.1 Pilas de codificador y decodificador

Codificador: El codificador se compone de una pila de N $= 6$ capas idénticas. Cada capa tiene dos subcapas. La primera es un mecanismo de auto-atención multi-cabeza, y la segunda es una red feed-forward simple, totalmente conectada en cuanto a la posición. Empleamos una conexión residual [11] alrededor de cada una de las dos subcapas, seguido de la normalización de la capa [1]. Es decir, la salida de cada subcapa es LayerNorm(x + Sublayer(x)), donde Sublayer(x es la función implementada por la subcapa en sí misma. Para facilitar estas conexiones residuales, todas las subcapas en el modelo, así como las capas de incrustación, producen salidas de dimensión d model

Decodificador: El decodificador también se compone de una pila de N $= 6$ capas idénticas. Además de las dos subcapas en cada capa del codificador, el decodificador inserta una tercera subcapa, que realiza la atención multi-cabeza sobre la salida de la pila del codificador. Similar al codificador, empleamos conexiones residuales alrededor de cada una de las subcapas, seguido de la normalización de la capa. También modificamos la subcapa de auto-atención en la pila del decodificador para evitar que las posiciones atiendan a las posiciones subsiguientes. Este enmascaramiento, combinado con el hecho de que las incrustaciones de salida se desplazan en una posición, asegura que las predicciones para la posición i puedan depender solo de las salidas conocidas en posiciones menores que i .

3.2 Atención

Una función de atención puede describirse como el mapeo de una consulta y un conjunto de pares clave-valor a una salida, donde la consulta, las claves, los valores y la salida son todos vectores. La salida se calcula como una suma ponderada

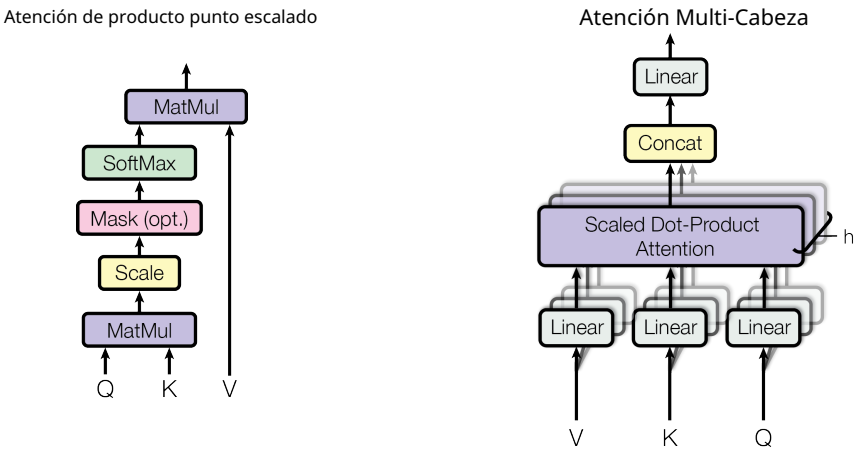

Figura 2: (izquierda) Atención de producto punto escalado. (derecha) La atención multi-cabeza consiste en varias capas de atención que se ejecutan en paralelo.

de los valores, donde el peso asignado a cada valor se calcula mediante una función de compatibilidad de la consulta con la clave correspondiente.

3.2.1 Atención de producto punto escalado

Llamamos a nuestra atención particular "Atención de producto punto escalado" (Figura 2). La entrada consiste en consultas y claves de dimensión d_k, y valores de dimensión d_v. Calculamos los productos punto de la consulta con todas las claves, dividimos cada uno por $\sqrt{d_k}$, y aplicamos una función softmax para obtener los pesos en los valores.

En la práctica, calculamos la función de atención en un conjunto de consultas simultáneamente, empaquetadas juntas en una matriz Q. Las claves y los valores también se empaquetan juntos en matrices K y V. Calculamos la matriz de salidas como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Las dos funciones de atención más utilizadas son la atención aditiva [2] y la atención de producto punto (multiplicativa). La atención de producto punto es idéntica a nuestro algoritmo, excepto por el factor de escala de $\frac{1}{\sqrt{d_k}}$. La atención aditiva calcula la función de compatibilidad utilizando una red feed-forward con una sola capa oculta. Si bien los dos son similares en complejidad teórica, la atención de producto punto es mucho más rápida y eficiente en espacio en la práctica, ya que se puede implementar utilizando código de multiplicación de matrices altamente optimizado.

Si bien para valores pequeños de d_k los dos mecanismos funcionan de manera similar, la atención aditiva supera a la atención de producto punto sin escala para valores más grandes de d_k [3]. Sospechamos que para valores grandes de d_k, los productos punto crecen mucho en magnitud, empujando la función softmax a regiones donde tiene gradientes extremadamente pequeños [4]. Para contrarrestar este efecto, escalamos los productos punto por $\frac{1}{\sqrt{d_k}}$.

3.2.2 Atención Multi-Cabeza

En lugar de realizar una sola función de atención con claves, valores y consultas de dimensión d_{model}, encontramos beneficioso proyectar linealmente las consultas, las claves y los valores h veces con diferentes proyecciones lineales aprendidas a dimensiones d_k, d_k y d_v, respectivamente. En cada una de estas versiones proyectadas de consultas, claves y valores, luego realizamos la función de atención en paralelo, produciendo d_v-dimensional

[4] Para ilustrar por qué los productos punto se hacen grandes, suponga que los componentes de q y k son aleatorios independientes variables con media 0 y varianza 1. Entonces su producto punto, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, tiene media 0 y varianza d_k.

valores de salida. Estos se concatenan y una vez más se proyectan, lo que da como resultado los valores finales, como se muestra en la Figura 2.

La atención multi-cabeza permite que el modelo atienda conjuntamente la información de diferentes subespacios de representación en diferentes posiciones. Con una sola cabeza de atención, el promedio inhibe esto.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{cabeza}_1, \ldots, \text{cabeza}_h)W^O$$
$$\text{donde cabeza} = \text{Attention}(QW_i^Q, KW_iK, VW_iV)$$

Donde las proyecciones son matrices de parámetros $W_i^Q \in R^{d_{\text{modelo}} \times d_k}$, $W_i^K \in R^{d_{\text{modelo}} \times d_k}$, $W_i^V \in R^{d_{\text{modelo}} \times d_v}$ y $W^O \in R^{hd_v \times d_{\text{modelo}}}$.

En este trabajo empleamos $h = 8$ capas de atención paralelas, o cabezas. Para cada una de estas usamos $d_k = d_v = d_{\text{modelo}}/h = 64$. Debido a la dimensión reducida de cada cabeza, el costo computacional total es similar al de la atención de una sola cabeza con dimensionalidad completa.

3.2.3 Aplicaciones de la atención en nuestro modelo

El transformador utiliza la atención multi-cabeza de tres maneras diferentes:

- En las capas de "atención codificador-decodificador", las consultas provienen de la capa decodificadora anterior, y las claves y valores de la memoria provienen de la salida del codificador. Esto permite que cada posición en el decodificador atienda a todas las posiciones en la secuencia de entrada. Esto imita los mecanismos típicos de atención codificador-decodificador en modelos de secuencia a secuencia como [38, 2, 9].
- El codificador contiene capas de auto-atención. En una capa de auto-atención, todas las claves, valores y las consultas provienen del mismo lugar, en este caso, la salida de la capa anterior en el codificador. Cada posición en el codificador puede atender a todas las posiciones en la capa anterior del codificador.
- Del mismo modo, las capas de auto-atención en el decodificador permiten que cada posición en el decodificador atienda a todas las posiciones en el decodificador hasta e incluyendo esa posición. Necesitamos evitar el flujo de información hacia la izquierda en el decodificador para preservar la propiedad auto-regresiva. Implementamos esto dentro de la atención de producto punto escalada enmascarando (estableciendo en $-\infty$) todos los valores en la entrada del softmax que corresponden a conexiones ilegales. Ver Figura 2.

3.3 Redes Feed-Forward Position-wise

Además de las subcapas de atención, cada una de las capas en nuestro codificador y decodificador contiene una red feed-forward totalmente conectada, que se aplica a cada posición por separado e idénticamente. Esto consiste en dos transformaciones lineales con una activación ReLU en el medio.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

Si bien las transformaciones lineales son las mismas en diferentes posiciones, utilizan diferentes parámetros de una capa a otra. Otra forma de describir esto es como dos convoluciones con tamaño de kernel 1. La dimensionalidad de la entrada y la salida es $d_{\text{modelo}} = 512$, y la capa interna tiene dimensionalidad $d_{ff} = 2048$.

3.4 Incrustaciones y Softmax

De manera similar a otros modelos de transducción de secuencia, utilizamos incrustaciones aprendidas para convertir los tokens de entrada y los tokens de salida en vectores de dimensión d_{modelo}. También utilizamos la transformación lineal aprendida habitual y la función softmax para convertir la salida del decodificador en probabilidades predichas del siguiente token. En nuestro modelo, compartimos la misma matriz de pesos entre las dos capas de incrustación y la transformación lineal pre-softmax, similar a [30]. En las capas de incrustación, multiplicamos esos pesos por $\sqrt{d_{\text{modelo}}}$.

Tabla 1: Longitudes máximas de ruta, complejidad por capa y número mínimo de operaciones secuenciales para diferentes tipos de capa. n es la longitud de la secuencia, d es la dimensión de la representación, k es el kernel tamaño de las convoluciones y r el tamaño de la vecindad en la autoatención restringida.

Tipo de capa	Complejidad por capa	Secuencial Operaciones	Longitud máxima de la ruta
Autoatención	$O(n^2 d)$	$O(1)$	$O(1)$
Recurrente	$O(n d^2)$	$O(n)$	$O(n)$
Convolucional	$O(k n d^2)$	$O(1)$	$O(\log_k(n))$
Autoatención (restringida)	$O(r n d)$	$O(1)$	$O(n/r)$

3.5 Codificación posicional

Dado que nuestro modelo no contiene recurrencia ni convolución, para que el modelo haga uso del orden de la secuencia, debemos inyectar información sobre la posición relativa o absoluta de los tokens en la secuencia. Para ello, agregamos "codificaciones posicionales" a las incrustaciones de entrada en la parte inferior de las pilas del codificador y decodificador. Las codificaciones posicionales tienen la misma dimensión d_{model} que las incrustaciones, de modo que las dos se pueden sumar. Hay muchas opciones de codificaciones posicionales, aprendidas y fijas [9].

En este trabajo, utilizamos funciones seno y coseno de diferentes frecuencias:

$$PE_{(pos,i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,i+1)} = \cos(pos/10000^{2i/d_{model}})$$

donde pos es la posición e i es la dimensión. Es decir, cada dimensión de la codificación posicional corresponde a una sinusoide. Las longitudes de onda forman una progresión geométrica desde 2π hasta $10000 \cdot 2\pi$. Nosotros elegimos esta función porque planteamos la hipótesis de que permitiría al modelo aprender fácilmente a atender por posiciones relativas, ya que para cualquier desplazamiento fijo k, PE_{pos+k} puede representarse como una función lineal de PE_{pos}.

También experimentamos con el uso de incrustaciones posicionales aprendidas [9] en su lugar, y descubrimos que las dos versiones produjeron resultados casi idénticos (ver la fila (E) de la Tabla 3). Elegimos la versión sinusoidal porque puede permitir que el modelo extrapole a longitudes de secuencia más largas que las encontradas durante el entrenamiento.

4 Por qué la autoatención

En esta sección comparamos varios aspectos de las capas de autoatención con las capas recurrentes y convolucionales que se utilizan comúnmente para mapear una secuencia de longitud variable de representaciones de símbolos $(x_1, ..., x_n)$ a otra secuencia de igual longitud $(z_1, ..., z_n)$, con $x_i, z_i \in \mathbb{R}^d$, como un oculto capa en un codificador o decodificador de transducción de secuencia típico. Motivando nuestro uso de la autoatención, consideramos tres desiderata.

Uno es la complejidad computacional total por capa. Otro es la cantidad de computación que se puede paralelizar, medida por el número mínimo de operaciones secuenciales requeridas.

El tercero es la longitud de la ruta entre las dependencias de largo alcance en la red. Aprender dependencias de largo alcance es un desafío clave en muchas tareas de transducción de secuencias. Un factor clave que afecta la capacidad de aprender tales dependencias es la longitud de las rutas que las señales hacia adelante y hacia atrás tienen que atravesar en la red. Cuanto más cortas sean estas rutas entre cualquier combinación de posiciones en las secuencias de entrada y salida, más fácil será aprender dependencias de largo alcance [12]. Por lo tanto, también comparamos la longitud máxima de la ruta entre dos posiciones de entrada y salida en redes compuestas por los diferentes tipos de capa.

Como se indica en la Tabla 1, una capa de autoatención conecta todas las posiciones con un número constante de operaciones ejecutadas secuencialmente, mientras que una capa recurrente requiere $O(n)$ operaciones secuenciales. En términos de complejidad computacional, las capas de autoatención son más rápidas que las capas recurrentes cuando la secuencia

la longitud n es menor que la dimensionalidad de la representación d , que es lo más común en el caso de representaciones de oraciones utilizadas por modelos de última generación en traducciones automáticas, como word-piece [38] y representaciones byte-pair [31]. Para mejorar el rendimiento computacional para tareas que involucran secuencias muy largas, la autoatención podría restringirse a considerar solo una vecindad de tamaño r en la secuencia de entrada centrada alrededor de la posición de salida respectiva. Esto aumentaría la longitud máxima de la ruta a O n/r . Planeamos investigar este enfoque más a fondo en el trabajo futuro.

Una sola capa convolucional con ancho de kernel k < n no conecta todos los pares de entrada y salida posiciones. Hacerlo requiere una pila de O n/k capas convolucionales en el caso de kernels contiguos, o O log k n)) en el caso de convoluciones dilatadas [18], aumentando la longitud de las rutas más largas entre dos posiciones cualesquiera en la red. Las capas convolucionales son generalmente más caras que las capas recurrentes, por un factor de k . Las convoluciones separables [6], sin embargo, disminuyen la complejidad considerablemente, a O k n d + n d 2). Incluso con k = n , sin embargo, la complejidad de una separable la convolución es igual a la combinación de una capa de autoatención y una capa de alimentación directa punto a punto, el enfoque que adoptamos en nuestro modelo.

Como beneficio adicional, la autoatención podría generar modelos más interpretables. Inspeccionamos las distribuciones de atención de nuestros modelos y presentamos y discutimos ejemplos en el apéndice. No solo los encabezados de atención individuales aprenden claramente a realizar diferentes tareas, sino que muchos parecen exhibir un comportamiento relacionado con la estructura sintáctica y semántica de las oraciones.

5 Entrenamiento

Esta sección describe el régimen de entrenamiento para nuestros modelos.

5.1 Datos de entrenamiento y procesamiento por lotes

Entrenamos con el conjunto de datos estándar WMT 2014 Inglés-Alemán que consta de aproximadamente 4.5 millones de pares de oraciones. Las oraciones se codificaron utilizando la codificación byte-pair [3], que tiene una fuente compartida-

vocabulario objetivo de aproximadamente 37000 tokens. Para inglés-francés, utilizamos el conjunto de datos WMT 2014 inglés -francés significativamente más grande que consta de 36 millones de oraciones y dividimos los tokens en un vocabulario de 32000 word-piece [38]. Los pares de oraciones se agruparon por longitud de secuencia aproximada. Cada entrenamiento

el lote contenía un conjunto de pares de oraciones que contenían aproximadamente 25000 tokens de origen y 25000 tokens de destino.

5.2 Hardware y Horario

Entrenamos nuestros modelos en una máquina con 8 GPU NVIDIA P100. Para nuestros modelos base que utilizan los hiperparámetros descritos en todo el documento, cada paso de entrenamiento tomó aproximadamente 0.4 segundos. Entrenamos los modelos base para un total de 100,000 pasos o 12 horas. Para nuestros modelos grandes (descritos en la línea inferior de la tabla 3), el tiempo de paso fue de 1.0 segundos. Los modelos grandes se entrenaron durante 300,000 pasos (3.5 días).

5.3 Optimizador

Usamos el optimizador Adam [20] con β 1 = 0. 9 β 2 = 0. 98 y ϵ = 10^{-9}. Variamos el aprendizaje tasa durante el curso del entrenamiento, de acuerdo con la fórmula:

$$\text{lrate} = d_{\text{modelo}}^{0.5} \cdot \min(\text{step\_num}^{-0.5}, \text{step\_num} \cdot \text{warmup\_steps}^{-1.5}) \quad (3)$$

Esto corresponde a aumentar la tasa de aprendizaje linealmente para los primeros warmup _ steps pasos de entrenamiento, y disminuirlo a partir de entonces proporcionalmente a la raíz cuadrada inversa del número de paso. Usamos warmup _ steps

5.4 Regularización

Empleamos tres tipos de regularización durante el entrenamiento:

Tabla 2: El Transformer logra mejores puntajes BLEU que los modelos de última generación anteriores en las pruebas newstest2014 de inglés a alemán e inglés a francés a una fracción del costo de entrenamiento.

Modelo	BLEU		Costo de entrenamiento (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (modelo base)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (grande)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Dropout Residual Aplicamos dropout [33] a la salida de cada subcapa, antes de que se agregue a la entrada de la subcapa y se normalice. Además, aplicamos dropout a las sumas de las incrustaciones y las codificaciones posicionales en las pilas del codificador y del decodificador. Para el modelo base, utilizamos una tasa de P drop

Suavizado de etiquetas Durante el entrenamiento, empleamos el suavizado de etiquetas del valor ϵ ls $= 0.1$ [36]. Esto perjudica la perplejidad, ya que el modelo aprende a estar más inseguro, pero mejora la precisión y el puntaje BLEU.

6 Resultados

6.1 Traducción automática

En la tarea de traducción de inglés a alemán WMT 2014, el modelo transformador grande (Transformer (grande) en la Tabla 2) supera a los mejores modelos informados anteriormente (incluidos los conjuntos) en más de 2 . 0 BLEU, estableciendo un nuevo puntaje BLEU de última generación de 28 . 4 . La configuración de este modelo es enumerado en la última línea de la Tabla 3. El entrenamiento tomó 3 . 5 días en 8 GPU P100. Incluso nuestro modelo base supera a todos los modelos y conjuntos publicados anteriormente, a una fracción del costo de entrenamiento de cualquiera de los modelos competitivos.

En la tarea de traducción de inglés a francés WMT 2014, nuestro modelo grande logra un puntaje BLEU de 41 . 0, superando a todos los modelos individuales publicados anteriormente, a menos de 1 4 del costo de entrenamiento del modelo de última generación anterior. El modelo Transformer (grande) entrenado para inglés a francés utilizó una tasa de abandono P drop

Para los modelos base, utilizamos un solo modelo obtenido promediando los últimos 5 puntos de control, que se escribieron a intervalos de 10 minutos. Para los modelos grandes, promediamos los últimos 20 puntos de control. Utilizamos la búsqueda de haz con un tamaño de haz de 4 y una penalización de longitud α $= 0.6$ [38]. Estos hiperparámetros se eligieron después de la experimentación en el conjunto de desarrollo. Establecemos la longitud máxima de salida durante la inferencia a la longitud de entrada + 50 , pero terminamos temprano cuando es posible [38].

La Tabla 2 resume nuestros resultados y compara nuestra calidad de traducción y costos de entrenamiento con otras arquitecturas de modelos de la literatura. Estimamos el número de operaciones de punto flotante utilizadas para entrenar un modelo multiplicando el tiempo de entrenamiento, el número de GPU utilizadas y una estimación de la capacidad de punto flotante de precisión simple sostenida de cada GPU [5].

6.2 Variaciones del modelo

Para evaluar la importancia de los diferentes componentes del Transformer, variamos nuestro modelo base de diferentes maneras, midiendo el cambio en el rendimiento en la traducción de inglés a alemán en el

[5] Utilizamos valores de 2.8, 3.7, 6.0 y 9.5 TFLOPS para K80, K40, M40 y P100, respectivamente.

Tabla 3: Variaciones en la arquitectura Transformer. Los valores no listados son idénticos a los del modelo base. Todas las métricas están en el conjunto de desarrollo de traducción de inglés a alemán, newstest2013. Las perplejidades listadas son por subpalabra, según nuestra codificación por pares de bytes, y no deben compararse con las perplejidades por palabra.

	N	d modelo	d ff	h	d k	d v	P drop	ϵ ls	entrenar pasos	PPL (desarrollo)	BLEU (desarrollo)	parámetros $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	incrustación posicional en lugar de sinusoides									4.92	25.7	
grande	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

conjunto de desarrollo, newstest2013. Utilizamos la búsqueda de haz como se describe en la sección anterior, pero sin promediar los puntos de control. Presentamos estos resultados en la Tabla 3.

En las filas (A) de la Tabla 3, variamos el número de cabezas de atención y las dimensiones de la clave y el valor de la atención, manteniendo constante la cantidad de cálculo, como se describe en la Sección 3.2.2. Si bien la atención de una sola cabeza es 0.9 BLEU peor que la mejor configuración, la calidad también disminuye con demasiadas cabezas.

En las filas (B) de la Tabla 3, observamos que reducir el tamaño de la clave de atención d k perjudica la calidad del modelo. Esto sugiere que determinar la compatibilidad no es fácil y que una función de compatibilidad más sofisticada que el producto punto puede ser beneficiosa. Además, observamos en las filas (C) y (D) que, como se esperaba, los modelos más grandes son mejores y la exclusión es muy útil para evitar el sobreajuste. En la fila (E) reemplazamos nuestra codificación posicional sinusoidal con incrustaciones posicionales aprendidas [9], y observamos resultados casi idénticos
resultados al modelo base.

6.3 Análisis de Constituyentes en Inglés

Para evaluar si el Transformer puede generalizarse a otras tareas, realizamos experimentos en el análisis de constituyentes en inglés. Esta tarea presenta desafíos específicos: la salida está sujeta a fuertes restricciones estructurales y es significativamente más larga que la entrada. Además, los modelos de secuencia a secuencia RNN no han podido alcanzar resultados de vanguardia en regímenes de datos pequeños [37].

Entrenamos un transformador de 4 capas con d modelo $= 1024$en la porción del Wall Street Journal (WSJ) del Penn Treebank [25], alrededor de 40K oraciones de entrenamiento. También lo entrenamos en un entorno semi-supervisado, utilizando los cuerpos de alta confianza y BerkleyParser más grandes de con aproximadamente 17 millones de oraciones [37]. Utilizamos un vocabulario de 16K tokens para la configuración solo WSJ y un vocabulario de 32K tokens para la configuración semi-supervisada.

Realizamos solo un pequeño número de experimentos para seleccionar la exclusión, tanto la atención como la residual (sección 5.4), las tasas de aprendizaje y el tamaño del haz en el conjunto de desarrollo de la Sección 22, todos los demás parámetros permanecieron sin cambios con respecto al modelo de traducción base de inglés a alemán. Durante la inferencia, nosotros

Tabla 4: El Transformer se generaliza bien al análisis sintáctico constituyente en inglés (los resultados están en la Sección 23 de WSJ)

Parser	Entrenamiento	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	Solo WSJ, discriminativo	88.3
Petrov et al. (2006) [29]	Solo WSJ, discriminativo	90.4
Zhu et al. (2013) [40]	Solo WSJ, discriminativo	90.4
Dyer et al. (2016) [8]	Solo WSJ, discriminativo	91.7
Transformer (4 capas)	Solo WSJ, discriminativo	91.3
Zhu et al. (2013) [40]	semi-supervisado	91.3
Huang & Harper (2009) [14]	semi-supervisado	91.3
McClosky et al. (2006) [26]	semi-supervisado	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervisado	92.1
Transformer (4 capas)	semi-supervisado	92.7
Luong et al. (2015) [23]	multi-tarea	93.0
Dyer et al. (2016) [8]	generativo	93.3

aumentó la longitud máxima de salida a la longitud de entrada + 300. Usamos un tamaño de haz de 21 y $\alpha = 0.3$ tanto para WSJ solamente como para el entorno semi-supervisado.

Nuestros resultados en la Tabla 4 muestran que, a pesar de la falta de ajuste específico de la tarea, nuestro modelo funciona sorprendentemente bien, obteniendo mejores resultados que todos los modelos informados anteriormente, con la excepción de la Gramática de Redes Neuronales Recurrentes [8].

En contraste con los modelos de secuencia a secuencia RNN [37], el Transformer supera a Berkeley-Parser [29] incluso cuando se entrena solo en el conjunto de entrenamiento WSJ de 40K oraciones.

7 Conclusión

En este trabajo, presentamos el Transformer, el primer modelo de transducción de secuencia basado completamente en la atención, reemplazando las capas recurrentes más comúnmente utilizadas en arquitecturas de codificador-decodificador con auto-atención multi-encabezada.

Para las tareas de traducción, el Transformer se puede entrenar significativamente más rápido que las arquitecturas basadas en capas recurrentes o convolucionales. Tanto en las tareas de traducción de inglés a alemán WMT 2014 como en las de inglés a francés WMT 2014, logramos un nuevo estado del arte. En la primera tarea, nuestro mejor modelo supera incluso a todos los conjuntos informados anteriormente.

Estamos entusiasmados con el futuro de los modelos basados en la atención y planeamos aplicarlos a otras tareas. Planeamos extender el Transformer a problemas que involucren modalidades de entrada y salida distintas del texto e investigar mecanismos de atención locales y restringidos para manejar de manera eficiente grandes entradas y salidas como imágenes, audio y video. Hacer que la generación sea menos secuencial es otro de nuestros objetivos de investigación.

El código que utilizamos para entrenar y evaluar nuestros modelos está disponible en https://github.com/tensorflow/tensor2tensor .

Agradecimientos Estamos agradecidos a Nal Kalchbrenner y Stephan Gouws por sus fructíferos comentarios, correcciones e inspiración.

Referencias

[1] Jimmy Lei Ba, Jamie Ryan Kiros y Geoffrey E Hinton. Normalización de capas. arXiv preprint arXiv:1607.06450 , 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho y Yoshua Bengio. Traducción automática neuronal por conjuntamente aprendiendo a alinear y traducir. CoRR , abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong y Quoc V. Le. Exploración masiva de redes neuronales arquitecturas de traducción automática. CoRR , abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong y Mirella Lapata. Redes de memoria a corto plazo para máquinas lectura. arXiv preprint arXiv:1601.06733 , 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, y Yoshua Bengio. Aprendizaje de representaciones de frases utilizando codificador-decodificador rnn para traducción automática estadística. CoRR , abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Aprendizaje profundo con convoluciones separables en profundidad. arXiv preprint arXiv:1610.02357 , 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho y Yoshua Bengio. Evaluación empírica de redes neuronales recurrentes cerradas en el modelado de secuencias. CoRR , abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros y Noah A. Smith. Red neuronal recurrente gramáticas. En Proc. de NAACL , 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats y Yann N. Dauphin. Convolu- aprendizaje secuencial a secuencial cional. arXiv preprint arXiv:1705.03122v2 , 2017.

[10] Alex Graves. Generando secuencias con redes neuronales recurrentes. arXiv preprint arXiv:1308.0850 , 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun. Aprendizaje residual profundo para im- reconocimiento de imágenes. En Actas de la Conferencia IEEE sobre Visión por Computador y Patrones Reconocimiento , páginas 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi y Jürgen Schmidhuber. Flujo de gradiente en redes recurrentes: la dificultad de aprender dependencias a largo plazo, 2001.

[13] Sepp Hochreiter y Jürgen Schmidhuber. Memoria a corto plazo larga. Computación neuronal , 9(8):1735–1780, 1997.

[14] Zhongqiang Huang y Mary Harper. Gramáticas PCFG de autoaprendizaje con anotaciones latentes entre idiomas. En Actas de la Conferencia de 2009 sobre Métodos Empíricos en Natural Procesamiento del lenguaje , páginas 832–841. ACL, agosto de 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer y Yonghui Wu. Explorando los límites del modelado del lenguaje. arXiv preprint arXiv:1602.02410 , 2016.

[16] Łukasz Kaiser y Samy Bengio. ¿Puede la memoria activa reemplazar la atención? En Avances en Neural Sistemas de procesamiento de información, (NIPS) , 2016.

[17] Łukasz Kaiser e Ilya Sutskever. Las GPU neuronales aprenden algoritmos. En Conferencia Internacional sobre Representaciones de Aprendizaje (ICLR) , 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves y Ko- ray Kavukcuoglu. Traducción automática neuronal en tiempo lineal. arXiv preprint arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang y Alexander M. Rush. Redes de atención estructuradas. En Conferencia Internacional sobre Representaciones de Aprendizaje , 2017.

[20] Diederik Kingma y Jimmy Ba. Adam: Un método para la optimización estocástica. En ICLR , 2015.

[21] Oleksii Kuchaiev y Boris Ginsburg. Trucos de factorización para redes LSTM. arXiv preprint arXiv:1703.10722 , 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou y Yoshua Bengio. Una incrustación de oraciones autoatenta estructurada. arXiv preprint arXiv:1703.03130 , 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals y Lukasz Kaiser. Multitarea aprendizaje secuencial a secuencial. arXiv preprint arXiv:1511.06114 , 2015.

[24] Minh-Thang Luong, Hieu Pham y Christopher D Manning. Enfoques efectivos para la atención- traducción automática neuronal basada. arXiv preprint arXiv:1508.04025 , 2015.

[25]Mitchell P Marcus, Mary Ann Marcinkiewicz y Beatrice Santorini. Construyendo un gran anotado corpus de inglés: The penn treebank. Lingüística computacional, 19(2):313–330, 1993.

[26]David McClosky, Eugene Charniak y Mark Johnson. Autoaprendizaje efectivo para el análisis sintáctico. En Actas de la Conferencia de Tecnología del Lenguaje Humano de la NAACL, Conferencia , Principal páginas 152–159. ACL, junio de 2006.

[27]Ankur Parikh, Oscar Täckström, Dipanjan Das y Jakob Uszkoreit. Una atención descomponible modelo. En Métodos empíricos en procesamiento del lenguaje natural, 2016.

[28]Romain Paulus, Caiming Xiong y Richard Socher. Un modelo reforzado profundo para la abstracción resumen. arXiv preprint arXiv:1705.04304 , 2017.

[29]Slav Petrov, Leon Barrett, Romain Thibaux y Dan Klein. Aprendizaje preciso, compacto, y anotación de árbol interpretable. En Actas de la 21ª Conferencia Internacional sobre Lingüística Computacional y 44ª Reunión Anual de la ACL, páginas 433–440. ACL, julio 2006.

[30]Ofir Press y Lior Wolf. Usando la incrustación de salida para mejorar los modelos de lenguaje. arXiv preprint arXiv:1608.05859 , 2016.

[31]Rico Sennrich, Barry Haddow y Alexandra Birch. Traducción automática neuronal de palabras raras con unidades de subpalabras. arXiv preprint arXiv:1508.07909 , 2015.

[32]Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, y Jeff Dean. Redes neuronales escandalosamente grandes: la capa de mezcla de expertos con puertas dispersas. arXiv preprint arXiv:1701.06538 , 2017.

[33]Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever y Ruslan Salakhutdinov. Dropout: una forma sencilla de evitar que las redes neuronales se sobreajusten. Revista de Máquina Investigación del aprendizaje, 15(1):1929–1958, 2014.

[34]Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston y Rob Fergus. Memoria de extremo a extremo redes. En C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama y R. Garnett, editores, Avances en los sistemas de procesamiento de información neuronal 28, páginas 2440–2448. Curran Associates, Inc., 2015.

[35]Ilya Sutskever, Oriol Vinyals y Quoc VV Le. Aprendizaje de secuencia a secuencia con redes neuronales redes. En Avances en los sistemas de procesamiento de información neuronal, páginas 3104–3112, 2014.

[36]Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens y Zbigniew Wojna. Repensando la arquitectura de inicio para la visión artificial. CoRR , abs/1512.00567, 2015.

[37]Vinyals & Kaiser, Koo, Petrov, Sutskever y Hinton. La gramática como lengua extranjera. En Avances en los sistemas de procesamiento de información neuronal, 2015.

[38]Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. El sistema de traducción automática neuronal de Google: cerrando la brecha entre la traducción humana y la automática. arXiv preprint arXiv:1609.08144 , 2016.

[39]Jie Zhou, Ying Cao, Xuguang Wang, Peng Li y Wei Xu. Modelos recurrentes profundos con conexiones de avance rápido para la traducción automática neuronal. CoRR , abs/1606.04199, 2016.

[40]Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang y Jingbo Zhu. Rápido y preciso análisis sintáctico constituyente de cambio-reducción. En Actas de la 51ª Reunión Anual de la ACL (Volumen 1: Documentos largos) , páginas 434–443. ACL, agosto de 2013.

Visualizaciones de atención de la capa 5 de entrada-entrada

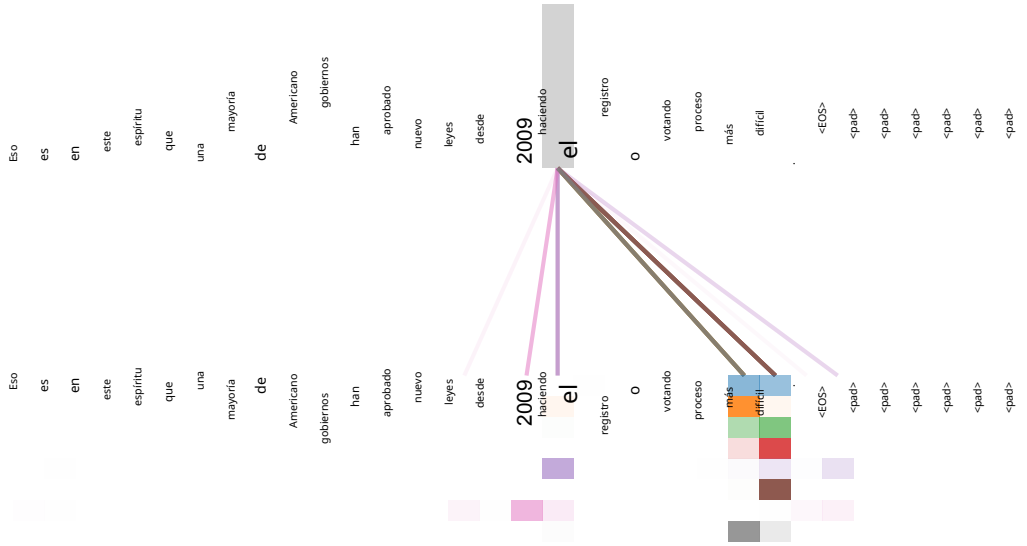

Figura 3: Un ejemplo del mecanismo de atención que sigue dependencias de larga distancia en la autoatención del codificador en la capa 5 de 6. Muchas de las cabezas de atención atienden a una dependencia distante del verbo 'haciendo', completando la frase 'haciendo...más difícil'. Atenciones aquí mostradas solo para la palabra 'haciendo'. Diferentes colores representan diferentes cabezas. Mejor visto en color.

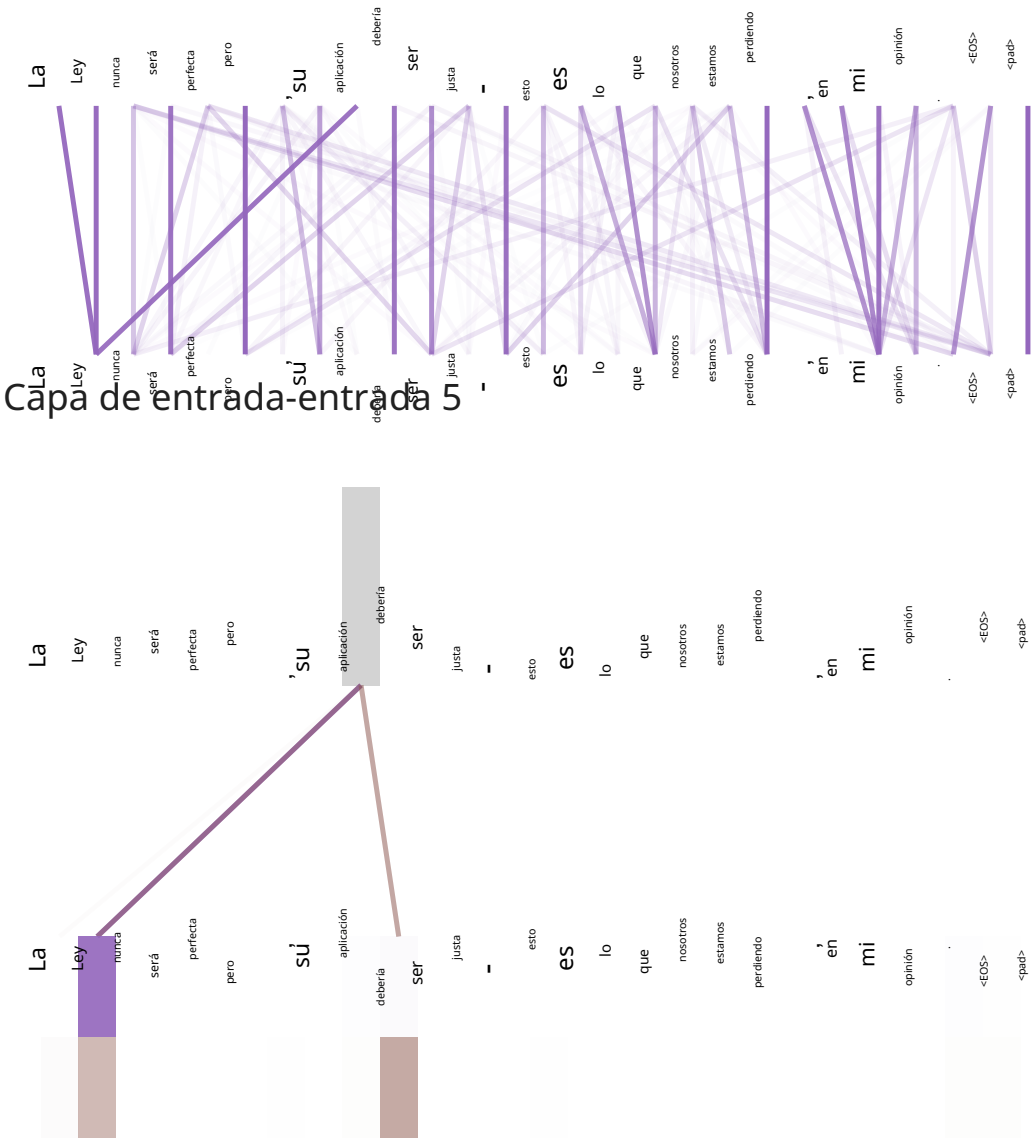

Figura 4: Dos cabezas de atención, también en la capa 5 de 6, aparentemente involucradas en la resolución de anáforas. Arriba: Atenciones completas para la cabeza 5. Abajo: Atenciones aisladas solo de la palabra 'su' para las cabezas de atención 5 y 6. Tenga en cuenta que las atenciones son muy nítidas para esta palabra.

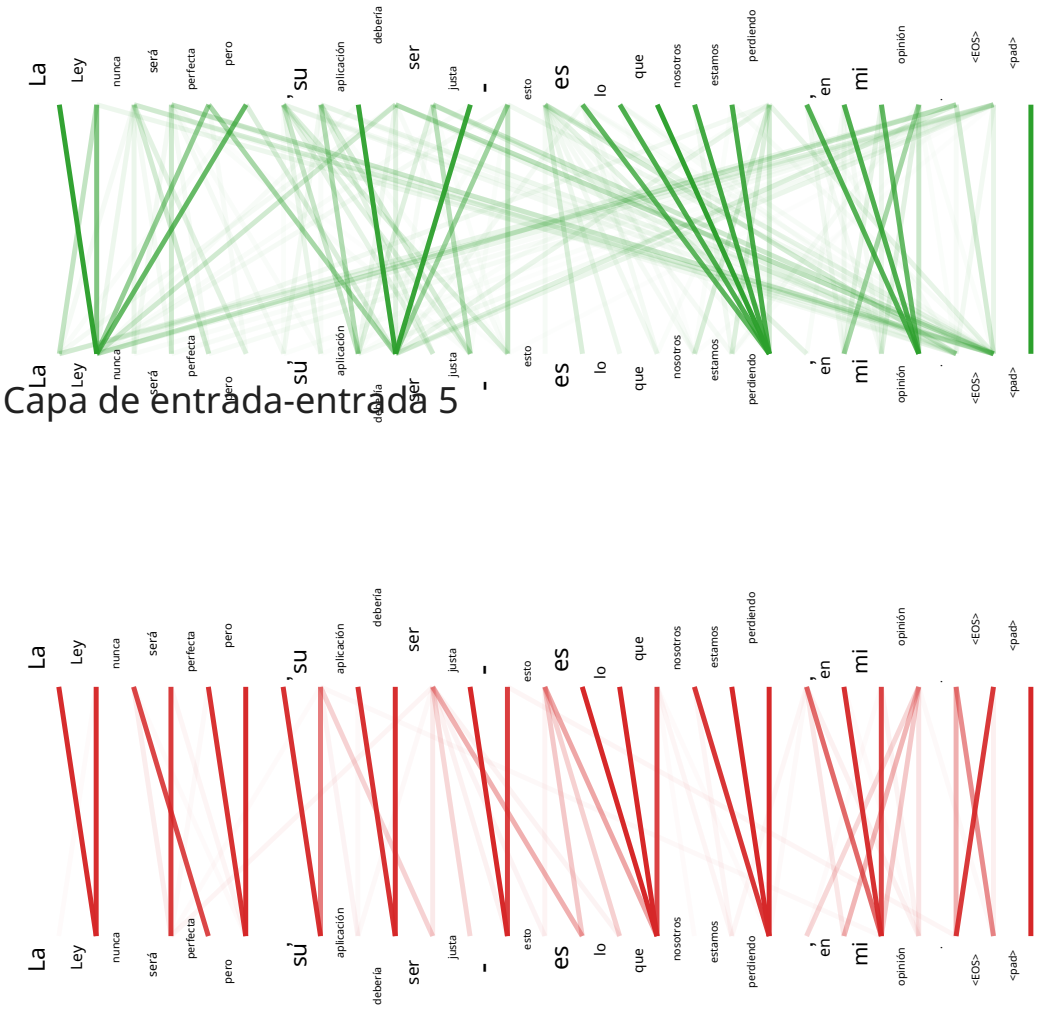

Figura 5: Muchas de las cabezas de atención exhiben un comportamiento que parece estar relacionado con la estructura de la oración. Damos dos ejemplos de este tipo arriba, de dos cabezas diferentes de la autoatención del codificador en la capa 5 de 6. Las cabezas claramente aprendieron a realizar diferentes tareas.