

Con la debida atribución se proporciona, Google otorga permiso para reproducir las tablas y figuras en este documento únicamente para uso en trabajos periodísticos o académicos.

La atención es todo lo que necesitas

A	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
L	Llion Jones* Google Research llion@google.com	Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu	Łukasz Kaiser* Google Brain lukaszkaiser@google.com	
Illia Polosukhin* ‡ illia.pолосухин@gmail.com				

Resumen

Los modelos de transducción de secuencia dominante se basan en complejas redes neuronales recurrentes o convolucionales que incluyen un encoder y un decodificador. Los mejores modelos de rendimiento también conectan el encoder 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, dispensando con recurrencia y convoluciones enteramente. Los experimentos en dos tareas de traducción automática muestran que estos modelos son superiores en calidad mientras que son más paralelos y requieren mucho menos tiempo para entrenar. Nuestro modelo alcanza 28.4 BLEU en la tarea de traducción de WMT 2014 Inglés a Alemán, mejorando sobre los mejores resultados existentes, incluyendo conjuntos, por más de 2 BLEU. En la tarea de traducción de inglés a francés WMT 2014 nuestro modelo establece una nueva puntuación de BLEU de última generación de 41.8 después de entrenar durante 3,5 días en ocho GPU, una pequeña fracción de los costos de entrenamiento de los mejores modelos de la literatura. Mostramos que el Transformer generaliza bien a otras tareas mediante la aplicación con éxito a la circunscripción inglesa analizando tanto con datos de entrenamiento grandes como limitados.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

1 Introducción

Las redes neuronales recurrentes, la memoria a corto plazo largo [13] y las redes neuronales cerradas en particular, se han establecido firmemente a medida que el estado de los enfoques artísticos en los problemas de modelado de secuencias y transducción, como el modelado de idiomas y la traducción automática [35, 2, 5]. Desde entonces, numerosos esfuerzos han seguido empujando los límites de los modelos de lenguaje recurrente y las arquitecturas de encoder-decoder [38, 24, 15].

Los modelos recurrentes suelen factorizar computación a lo largo de las posiciones de símbolo de las secuencias de entrada y salida. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t , as a function of the previous hidden state h_{t-1} and the input for position t . Esta naturaleza inherentemente secuencial excluye la paralelización dentro de los ejemplos de capacitación, que se vuelve crítica a largos períodos de secuencia, ya que las limitaciones de memoria limitan el batido en los ejemplos. La labor reciente ha logrado mejoras significativas en la eficiencia computacional mediante trucos de factorización [21] y cálculo condicional [32], al tiempo que ha mejorado el rendimiento de los modelos en caso de estos últimos. Sin embargo, sigue existiendo la limitación fundamental de la computación secuencial.

Los mecanismos de atención se han convertido en parte integral de modelos de secuencias convincentes y de transición en diversas tareas, permitiendo modelar dependencias sin tener en cuenta su distancia en las secuencias de entrada o salida [2, 19]. Sin embargo, en todos los casos [27], esos mecanismos de atención se utilizan conjuntamente con una red recurrente.

En este trabajo proponemos al Transformer, una arquitectura modelo que esquiva la recurrencia y, en cambio, dependemos totalmente de un mecanismo de atención para atraer dependencias globales entre la entrada y la salida. El transformador permite una mayor parallelización y puede llegar a un nuevo estado del arte en calidad de traducción después de ser entrenado por tan sólo doce horas en ocho GPUs P100.

2 Antecedentes

El objetivo de reducir la computación secuencial también constituye la base de la GPU Neural Extended [16], ByteNet [18] y ConvS2S [9], todos los cuales utilizan las redes neuronales convolucionales como bloque de construcción básico, computando representaciones ocultas en paralelo para todas las posiciones de entrada y salida. En estos modelos, el número de operaciones requeridas 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 más difícil aprender dependencias entre posiciones distantes [12]. En el Transformer esto se reduce a un número constante de operaciones, aunque al costo de una resolución eficaz reducida debido a la promediación de posiciones ponderadas en la atención, un efecto que contrarresta con la atención Multi-Head como se describe en la sección 3.2.

La autoatención, a veces llamada intraatención 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 diversas tareas, entre ellas la comprensión de la lectura, la resumenización abstractiva, la implicación textual y las representaciones de frases independientes de la tarea de aprendizaje [4, 27, 28, 22].

Las redes de memoria de extremo a extremo se basan en un mecanismo de atención recurrente en lugar de la recurrencia alineada de secuencia y se han demostrado para realizar bien en tareas de respuesta de preguntas sencillas y de modelado de idiomas [34].

Sin embargo, al mejor de nuestros conocimientos, el Transformer es el primer modelo de transducción que se basa enteramente en la autoatención para calcular las representaciones de su entrada y salida sin utilizar RNNs alineados con secuencia o convolución. En las secciones siguientes, describiremos al Transformador, motivaremos la autoatención y discutiremos sus ventajas sobre modelos como [17, 18] y [9].

3 Modelo de Arquitectura

Los modelos de transducción de secuencias neuronales más competitivos tienen una estructura de encoder-decoder [5, 2, 35]. Aquí, el encoder mapea una secuencia de entrada de representaciones de símbolos (x_1, \dots, x_n) a una secuencia de representaciones continuas $\mathbf{z} = (z_1, \dots, z_n)$. Dado \mathbf{z} , el decodificador genera entonces una secuencia de salida (y_1, \dots, y_m) de símbolos un elemento a la vez. En cada paso el modelo es auto-regresivo [10], consumiendo los símbolos previamente generados como entrada adicional al generar el siguiente.

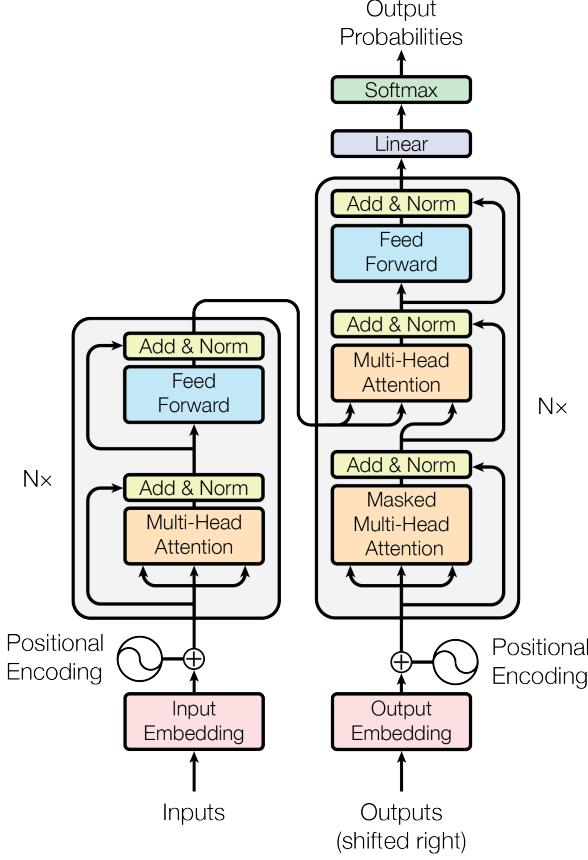


Figura 1: El transformador - arquitectura modelo.

El transformador sigue esta arquitectura global utilizando autoatención apilada y capas de punto, totalmente conectadas para el encoder y decodificador, mostrados en las mitades izquierda y derecha de la Figura 1, respectivamente.

3.1 Encoder and Decoder Stacks

Codificador: El encoder se compone de una pila de $N = 6$ capas idénticas. Cada capa tiene dos capas. El primero es un mecanismo de auto-atención multicabezas, y el segundo es una red simple, de posición- inteligente totalmente conectado. Empleamos una conexión residual [11] alrededor de cada una de las dos subcapas, seguida de normalización de capas [1]. Es decir, la salida de cada subcapa es $\text{LayerNorm}(x + \text{Sublayer}(x))$, donde $\text{Sublayer}(x)$ es la función implementada por la sub-capa. Para facilitar estas conexiones residuales, todas las subcapas del modelo, así como las capas de embedding, producen salidas de dimensión $d_{\text{model}} = 512$.

Decodificador: El decodificador también se compone de una pila de $N = 6$ capas idénticas. Además de las dos subcapas en cada capa de encoder, el decodificador inserta una tercera subcapa, que realiza la atención multicabezas sobre la salida de la pila de encoder. Al igual que el encoder, emplea conexiones residuales alrededor de cada una de las subcapas, seguidas de normalización de capas. También modificamos la subcapa de autoatención en la pila de decodificador para evitar que las posiciones asistan a posiciones posteriores. Este enmascaramiento, combinado con el hecho de que las incrustaciones de salida se compensan por una posición, asegura que las predicciones para la posición i sólo pueden depender de los productos conocidos en posiciones inferiores a i .

3.2 Atención

Una función de atención se puede describir como mapear una consulta y un conjunto de pares de valor clave a una salida, donde la consulta, claves, valores y salida son todos vectores. La salida se calcula como una suma ponderada

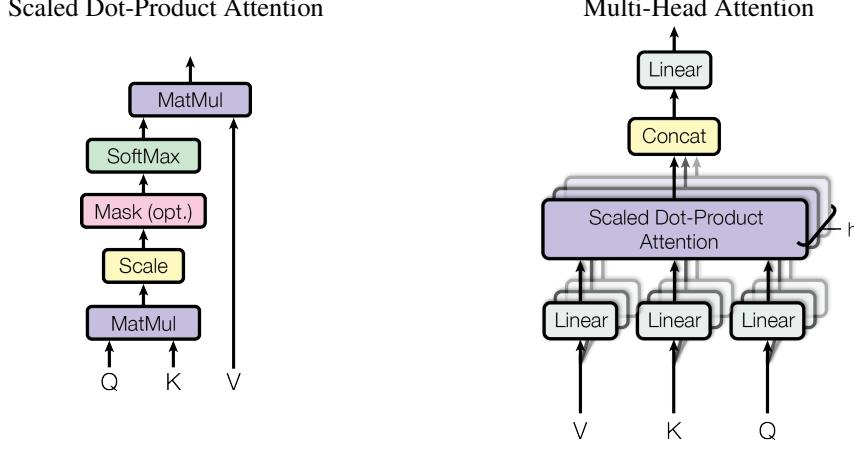


Figure 2: (left) Scaled Dot-Product Atención. (right) Multi-Head La atención consiste en varias capas de atención que se ejecutan en paralelo.

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

3.2.1 Atención de productos de punto escalada

Llamamos nuestra atención particular "Atención de Dot-Producto Escalada" (Figura 2). La entrada consiste en consultas y claves de dimensión d_k , y valores de dimensión d_v . Computamos los productos de puntos de la consulta con todas las teclas, dividir cada uno por $\sqrt{d_k}$, y aplicar una función softmax para obtener los pesos en los valores.

En la práctica, computamos la función de atención en un conjunto de consultas simultáneamente, empacadas en una matriz Q . Las llaves y los valores también están empacados en matrices K y V . Computamos la matriz de salidas como:

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

Las dos funciones de atención más utilizadas son la atención aditiva [2], y la atención del producto de puntos (multi- plicativo). La atención del producto de puntos es idéntica a nuestro algoritmo, excepto por el factor de escalado de $\frac{1}{\sqrt{d_k}}$. La atención aditiva calcula la función de compatibilidad utilizando una red de alimentación hacia adelante con una sola capa oculta. Mientras que los dos son similares en la complejidad teórica, la atención del producto de puntos es mucho más rápida y más eficiente en el espacio en la práctica, ya que se puede implementar utilizando código de multiplicación de matriz altamente optimizado.

Mientras que para los pequeños valores de d_k los dos mecanismos funcionan de manera similar, la atención aditiva supera la atención del producto punto sin escalar para valores más grandes de d_k [3]. Sospechamos que para grandes valores de d_k , los productos de punto crecen en gran magnitud, empujando la función softmax en regiones donde tiene gradientes extremadamente pequeños⁴. Para contrarrestar este efecto, escalamos los productos de puntos por $\frac{1}{\sqrt{d_k}}$.

3.2.2 Atención Multi-Head

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

⁴To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k .

valores de salida. Estos están concatenados y una vez más proyectados, dando lugar a los valores finales, como se describe en la Figura 2.

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

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Donde las proyecciones son matrices de parámetro $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ y $W^O \in \mathbb{R}^{h d_v \times d_{\text{model}}}$.

En este trabajo empleamos $h = 8$ capas de atención paralelas, o cabezas. Para cada uno de estos utilizamos $d_k = d_v = d_{\text{model}}/h = 64$. Debido a la dimensión reducida de cada cabeza, el costo total computacional es similar al de la atención de un solo cabeza con dimensión completa.

3.2.3 Aplicaciones de atención en nuestro modelo

El transformador utiliza la atención multicabezas de tres maneras diferentes:

- En capas de "atención del encoder", las consultas vienen de la capa anterior de decodificador, y las claves de memoria y los valores provienen de la salida del encoder. Esto permite que cada posición en el decodificador asista a todas las posiciones en la secuencia de entrada. Esto imita los mecanismos típicos de atención del encoder-decodificador en modelos secuencia-a secuencia como [38, 2, 9].
- El encoder contiene capas de autoatención. En una capa de autoatención todas las teclas, valores y consultas vienen del mismo lugar, en este caso, la salida de la capa anterior en el encoder. Cada posición en el encoder puede asistir a todas las posiciones en la capa anterior del encoder.
- Del mismo modo, las capas de autoatención en el decodificador permiten que cada posición en el decodificador asista 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 este interior de la atención de punto-producto escalada enmascarando (ajustando a $-\infty$) todos los valores en la entrada del softmax que corresponden a conexiones ilegales. Véase la Figura 2.

3.3 Redes Feed-Forward basadas en la posición

Además de las subcapas de atención, cada una de las capas de nuestro encoder y decodificador contiene una red de alimentación completa conectada, que se aplica a cada posición por separado e idénticamente. Esto consiste en dos transformaciones lineales con una activación de ReLU entre sí.

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

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

3.4 Embeddings and Softmax

De manera similar a otros modelos de transducción de secuencias, utilizamos incrustaciones aprendidas para convertir los símbolos de entrada y salida a vectores de dimensión d_{model} . También utilizamos la función transformación lineal usualmente aprendida y softmax para convertir la salida de decodificador para predecir las probabilidades de contacto siguiente. En nuestro modelo, compartimos la misma matriz de peso entre las dos capas de embedding y la transformación lineal pre-softmax, similar a [30]. En las capas de incrustación, multiplicamos esos pesos por $\sqrt{d_{\text{model}}}$.

Cuadro 1: Longitudes máximas, complejidad por capa y número mínimo de operaciones secuenciales para diferentes tipos de capas. n es la longitud de la secuencia, d es la dimensión de representación, k es el tamaño del núcleo de las convoluciones y r el tamaño del vecindario en autoatención resaltada.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Codificación de la posición

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 las fichas en la secuencia. Para ello, agregamos "encodings posicionales" a las incrustaciones de entrada en los fondos de las pilas de encoder y decodificador. Las codificaciones de posición tienen la misma dimensión d_{model} que las incrustaciones, para que las dos puedan ser resumidas. Hay muchas opciones de codificación posicional, aprendida y fijada [9].

En este trabajo utilizamos funciones sine y coseno de diferentes frecuencias:

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

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

donde pos es la posición y i es la dimensión. Es decir, cada dimensión de la codificación posicional corresponde a un sinusoide. Las longitudes de onda forman una progresión geométrica de 2π a $10000 \cdot 2\pi$. Elegimos esta función porque hipotetizamos que permitiría al modelo aprender fácilmente a asistir por posiciones relativas, ya que para cualquier compensación fija k , PE_{pos+k} puede ser representado como una función lineal de PE_{pos} .

También experimentamos con el uso de embeddings posicionales aprendidos [9] en su lugar, y encontramos que las dos versiones produjeron resultados casi idénticos (véase la fila Tabla 3 (E)). Elegimos la versión sinusoidal porque puede permitir que el modelo extrapolar 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 a las capas recurrentes y convolucionales utilizadas comúnmente para mapear una secuencia de longitud variable de las representaciones de los símbolos (x_1, \dots, x_n) a otra secuencia de igual longitud (z_1, \dots, z_n) , con $x_i, z_i \in \mathbb{R}^d$, como una capa oculta en un encoder de transducción de secuencia típica o decodificador. Motivando nuestro uso de la autoatención consideramos tres desiderata.

Una es la complejidad computacional total por capa. Otra es la cantidad de cálculo que puede ser paralelizada, medida por el número mínimo de operaciones secuenciales requeridas.

El tercero es la longitud del camino entre dependencias de largo alcance en la red. Aprender dependencias de largo alcance es un reto clave en muchas tareas de transducción de secuencias. Un factor clave que afecta la capacidad de aprender tales dependencias es la longitud de los caminos hacia adelante y las señales atrasadas tienen que atravesar en la red. Cuanto más cortos estos caminos entre cualquier combinación de posiciones en las secuencias de entrada y salida, más fácil es aprender dependencias de largo alcance [12]. Por lo tanto, también comparamos la longitud de la ruta máxima entre las dos posiciones de entrada y salida en redes compuestas por los diferentes tipos de capas.

Como se señala 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

longitud n es menor que la dimensionalidad de representación d , que es más a menudo el caso con representaciones de frases utilizadas por modelos de última generación en traducciones de máquina s, tales como las representaciones de palabra-pie [38] y byte-pair [31]. Para mejorar el rendimiento computacional para tareas que implican secuencias muy largas, la autoatención podría limitarse a considerar sólo un vecindario de tamaño r en la secuencia de entrada centrada alrededor de la posición de salida respectiva. Esto aumentaría la longitud máxima del camino a $O(n/r)$. Planeamos seguir investigando este enfoque en futuros trabajos.

Una sola capa convocional con ancho del núcleo $k < n$ no conecta todos los pares de posiciones de entrada y salida. Hacerlo requiere una pila de $O(n/k)$ capas convolutivas en el caso de núcleos con tiguos, o $O(\log_k(n))$ en el caso de convoluciones dilatadas [18], aumentando la longitud de los ca minos más largos entre las dos posiciones en la red. Las capas convolutivas son generalmente más caras que las capas recurrentes, por un factor de k . Convoluciones estables [6], sin embargo, disminuir la complejidad considerablemente, a $O(k \cdot n \cdot d + n \cdot d^2)$. Incluso con $k = n$, sin embargo, la complejidad de una convolución separable es igual a la combinación de una capa de auto-atención y una capa de alimentación-acción de punta, el enfoque que tomamos en nuestro modelo.

Como beneficio secundario, la autoatención podría producir modelos más interpretables. Inspeccionamos las distribuciones de atención de nuestros modelos y presentamos y discutimos ejemplos en el apéndice. No sólo los jefes de atención individual aprenden claramente a realizar diferentes tareas, muchos parecen mostrar comportamiento relacionado con la estructura sintáctica y semántica de las oraciones.

5 Formación

Esta sección describe el régimen de entrenamiento para nuestros modelos.

5.1 Datos de capacitación y batido

Entrenamos en el conjunto de datos estándar WMT 2014 Inglés-German que consiste en unos 4,5 m illones de pares de frases. Las sentencias fueron codificadas mediante la codificación byte-pair [3], que tiene un vocabulario de origen compartido de aproximadamente 37000 fichas. Para inglés-francés, utilizamos el conjunto de datos WMT 2014 de tamaño significativo que consta de oraciones de 36M y fichas divididas en un vocabulario de 32000 palabras [38]. Los pares de sentence fueron batidos juntos por longitud de secuencia aproximada. Cada lote de entrenamiento contenía un conjunto de pares de frases que contenían aproximadamente 25000 fichas de origen y 25000 fichas de destino.

5.2 Hardware and Schedule

Entrenamos nuestros modelos en una máquina con 8 GPUs NVIDIA P100. Para nuestros modelos de base utilizando los hiperparametros descritos a lo largo del papel, cada paso de entrenamiento tomó alrededor de 0.4 segundos. Entrenamos los modelos base para un total de 100.000 pasos o 12 horas. Para nuestros grandes modelos (descritos en la línea inferior de la tabla 3), el tiempo del paso fue de 1.0 segundos. Los grandes modelos fueron entrenados para 300.000 pasos (3,5 días).

5.3 Optimizador

Usamos el optimizador Adán [20] con $\beta_1 = 0.9$, $\beta_2 = 0.98$ y $\epsilon = 10^{-9}$. Variamos la tasa de aprendizaje durante el curso de la formación, según la fórmula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \quad (3)$$

Esto corresponde a aumentar la tasa de aprendizaje linealmente para los primeros $warmup_steps$ pasos de entrenamiento, y disminuirla posteriormente proporcionalmente a la raíz inversa cuadrada del número de paso. Usamos 4000.

5.4 Regularización

Empleamos tres tipos de regularización durante el entrenamiento:

Tabla 2: El transformador logra mejores puntuaciones de BLEU que los modelos de última generación en inglés-alemán y inglés-a-francés 2014 pruebas a una fracción del coste de entrenamiento.

Model	BLEU		Training Cost (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 (base model)	27.3	38.1		$3.3 \cdot 10^{18}$
Transformer (big)	28.4	41.8		$2.3 \cdot 10^{19}$

Residual Dropout Aplicamos el deserción [33] a la salida de cada subcapa, antes de que se añada a la entrada de subcapa y se normalice. Además, aplicamos el abandono a las sumas de las incrustaciones y las codificaciones posicionales en las pilas de encoder y decodificador. Para el modelo base, utilizamos una tasa de $P_{drop} = 0.1$.

Label Smoothing Durante el entrenamiento, empleamos etiqueta suavizar el valor $\epsilon_{ls} = 0.1$ [36]. Esto duele la perplejidad, ya que el modelo aprende a ser más inseguro, pero mejora la precisión y la puntuación de BLEU.

6 Resultados

Traducción de máquinas

En la tarea de traducción de WMT 2014 Inglés a Alemán, el modelo de transformador grande (Transformer (big) en la tabla 2) supera los mejores modelos previamente reportados (incluidos los conjuntos) por más de 2.0 BLEU, estableciendo una nueva puntuación de BLEU de última generación de 28.4. La configuración de este modelo se enumera en la línea inferior de la tabla 3. El entrenamiento tomó 3.5 días en 8 GPUs P100. Incluso nuestro modelo base supera 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 WMT 2014 Inglés a Francés, nuestro gran modelo logra una puntuación de BLEU de 41.0, superando todos los modelos individuales publicados anteriormente, a menos de 1/4 el coste de entrenamiento del modelo anterior de última generación. El modelo Transformer (big) entrenado para el inglés-a-francés utilizó la tasa de deserción $P_{drop} = 0.1$, en lugar de 0.3.

Para los modelos de base, usamos un modelo único obtenido por medio de los últimos 5 puestos de control, que fueron escritos a intervalos de 10 minutos. Para los grandes modelos, promediamos los últimos 20 puestos de control. Usamos la búsqueda de haz con un tamaño de haz de 4 y la pena de longitud $\alpha = 0.6$ [38]. Estos hiperparámetros fueron elegidos después de la experimentación en el conjunto de desarrollo. Ponemos la longitud máxima de salida durante la inferencia a la longitud de entrada + 50, pero terminamos pronto cuando sea posible [38].

En el cuadro 2 se resumen nuestros resultados y se comparan nuestros costos de calidad de traducción y capacitación a otras arquitecturas modelo de la literatura. Estimamos el número de operaciones de puntos flotantes utilizadas para formar un modelo multiplicando el tiempo de entrenamiento, el número de GPU utilizados, y una estimación de la capacidad flotante de puntos de precisión sostenida de cada GPU⁵.

Variaciones modelo 6.2

Para evaluar la importancia de los diferentes componentes del Transformer, hemos variado nuestro modelo base de diferentes maneras, midiendo el cambio de rendimiento en la traducción Inglés-A-German en la

⁵We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

Tabla 3: Variaciones sobre la arquitectura Transformer. Los valores no listados son idénticos a los del modelo base. Todas las métricas están en el set de desarrollo de traducción Inglés-A-Alemania, Newstest2013. Las perplexidades enumeradas son por palabra, según nuestra codificación byte-pair , y no deben compararse con perplexidades por palabra.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
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
							0.0			5.77	24.6	
(D)							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
	(E)									4.92	25.7	
big	6	1024	4096	16			0.3		300K	4.33	26.4	213

set de desarrollo, newstest2013. Usamos la búsqueda de haz como se describe en la sección anterior, pero no hay un promedio de puntos de control. Presentamos estos resultados en la tabla 3.

En la tabla 3 filas (A), cambiamos el número de cabezales de atención y las dimensiones clave y de valor de la atención, manteniendo la cantidad de computación constante, como se describe en la sección 3.2.2. Aunque la atención de un solo cabeza es 0.9 BLEU peor que el mejor ajuste, la calidad también baja con demasiadas cabezas.

En la tabla 3 filas (B), observamos que reducir el tamaño de la atención clave d_k duele la calidad de 1 modelo. Esto sugiere que determinar la compatibilidad no es fácil y que una función de compatibilidad más sofisticada que el producto del punto puede ser beneficioso. Observamos además en las filas (C) y (D) que, como se esperaba, los modelos más grandes son mejores, y el abandono es muy útil para evitar el exceso de ajuste. En fila (E) reemplazamos nuestra codificación posicional sinusoidal con embeddings posicionales aprendidos [9], y observamos resultados casi idénticos al modelo base.

6.3 Programación de circunscripciones en inglés

Evaluando si el Transformer puede generalizarse a otras tareas que realizamos experimentos en el análisis de la circunscripción inglesa. Esta tarea presenta retos específicos: la producción está sujeta a fuertes limitaciones estructurales y es significativamente más larga que la entrada. Furthermore, RN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

Entrenamos un transformador de 4 capas con $d_{model} = 1024$ en la porción Wall Street Journal (WSJ) del Penn Treebank [25], alrededor de 40K oraciones de entrenamiento. También lo entrenamos en un entorno semi-supervisado, utilizando la mayor alta confianza y BerkleyParser corpora de aproximadamente 17M oraciones [37]. Usamos un vocabulario de 16K tokens para el ajuste WSJ y un vocabulario de 32K tokens para el ajuste semi-supervisado.

Hemos realizado sólo un pequeño número de experimentos para seleccionar el abandono, tanto la atención como el residual (sección 5.4), los índices de aprendizaje y el tamaño de la viga en el conjunto de desarrollo de la Sección 22, todos los demás parámetros permanecieron inalterados del modelo de traducción de base Inglés-A-Alemania. Durante la inferencia, nosotros

Cuadro 4: El transformador generaliza bien la parización de la circunscripción inglesa (Los resultados están en la sección 23 de WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

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

Nuestros resultados en la Tabla 4 muestran que, a pesar de la falta de ajuste específico de tareas nuestro modelo funciona perfectamente bien, dando mejores resultados que todos los modelos previamente reportados con la excepción de la Red Neural Recurrent Grammar [8].

En contraste con los modelos de secuencia a secuencia de RNN [37], el Transformer supera al Berkely-Parser [29] incluso cuando el entrenamiento sólo en el conjunto de entrenamiento WSJ de frases de 40K.

7 Conclusiones

En este trabajo, presentamos el Transformer, el primer modelo de transducción de secuencia basado enteramente en la atención, reemplazando las capas recurrentes más utilizadas en las arquitecturas de encoder-decodificador con autoatención multicabezada.

Para tareas de traducción, el transformador puede ser entrenado considerablemente más rápido que las arquitecturas basadas en capas recurrentes o convolutivas. Tanto en WMT 2014 tareas de traducción de inglés a alemán como WMT 2014 en inglés a francés, logramos un nuevo estado del arte. En la primera tarea nuestro mejor modelo supera incluso todos los conjuntos previamente reportados.

Nos entusiasma el futuro de los modelos basados en la atención y el plan para aplicarlos a otras tareas. Planeamos extender el Transformador a problemas que implican modalidades de entrada y salida distintas del texto e investigar mecanismos locales de atención restringida para manejar eficientemente grandes insumos y salidas como imágenes, audio y vídeo. Hacer que la generación sea menos secuencial es otro objetivo de investigación nuestro.

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

Reconocimientos Estamos agradecidos a Nal Kalchbrenner y Stephan Gouws por sus fructíferas observaciones, correcciones e inspiración.

Referencias

- [1] Jimmy Lei Ba, Jamie Ryan Kiros y Geoffrey E Hinton. Normalización de capas. 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, y Yoshua Bengio. Traducción de máquina neuronal mediante el aprendizaje conjunto para alinear y traducir. *CoRR*, abs/1409.0473, 2014.
- [3] Denny Britz, Anna Goldie, Minh-Thang Luong y Quoc V. Le. Exploración masiva de arquitecturas de traducción de máquinas neuronales. *CoRR*, abs/1703.03906, 2017.
- [4] Jianpeng Cheng, Li Dong y Mirella Lapata. Redes de memoria a corto plazo para la lectura de máquinas. 2016.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk y Yoshua Bengio. Aprendizaje de frases representaciones usando un encoder-decoder para traducción automática estadística. *CoRR*, abs/1406.1078, 2014.

François Fleuret. Xception: Aprendizaje profundo con revoluciones separables. 2016.

[7] Junyoung Chung, Çağlar Gülcabay, Kyunghyun Cho, y Yoshua Bengio. Evaluación empírica de redes neuronales recidivas cerradas en el modelado de secuencias. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, y Noah A. Smith. Gramáticas de red neuronal recurrente. En *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats y Yann N. Dauphin. Convolución secuencial a la secuencia de aprendizaje. 2017.

Alex Graves. Generando secuencias con redes neuronales recurrentes. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. Deep residual learning for image recognition. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 770–778, 2016.

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

[13] Sepp Hochreiter y Jürgen Schmidhuber. Memoria a corto plazo. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang y Mary Harper. Gramática PCFG auto-entrenadora con anotaciones latentes a través de idiomas. En las páginas 832–841. ACL, agosto de 2009.

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

[16] Loukas Kaiser y Samy Bengio. ¿Puede la memoria activa reemplazar la atención? En *Advances in Neural Information Processing Systems*, (NIPS), 2016.

[17] Loukas Kaiser e Ilya Sutskever. Las GPU neuronales aprenden algoritmos. En *International Conference on Learning Representations (ICLR)*, 2016.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, y Koray Kavukcuoglu. Traducción de máquina 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 *International Conference on Learning Representations*, 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 sentencia autoatentativa estructurada que incrusta. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, y Lukasz Kaiser. Secuencia multitarea para el aprendizaje de secuencias. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham y Christopher D Manning. Enfoques eficaces para la traducción de la máquina neuronal basada en la atención. *arXiv preprint arXiv:1508.04025*, 2015.

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

David McClosky, Eugene Charniak y Mark Johnson. Auto-entrenamiento eficaz para la par. En las páginas 152-159. ACL, junio de 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, y Jakob Uszkoreit. Un modelo de atención des compuesta. En *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong y Richard Socher. Un modelo reforzado profundo para la resumenización abstracta. 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, y Dan Klein. Aprender anotación de árboles precisa, compacta e interpretable. En las páginas 433-440. ACL, julio de 2006.

[30] Ofir Press y Lior Wolf. Utilizando la incorporación de salida para mejorar los modelos de lenguaje. 2016.

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

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, y Jeff Dean. Redes neuronales extremadamente grandes: La capa de mezcla de expertos escasamente pegada. 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, y Ruslan Salakhutdinov. Caída: una manera sencilla de evitar que las redes neuronales se superen. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston y Rob Fergus. Redes de memoria final a extremo. En C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama y R. Garnett, editores, *Advances in Neural Information Processing Systems* 28, págs. 2440 a 2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, y Quoc VV Le. Secuencia para aprender secuencia con redes neuronales. En *Advances in Neural Information Processing Systems*, páginas 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, y Zbigniew Wojna. Re-pensar la arquitectura de la creación para la visión de la computadora. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals " Kaiser, Koo, Petrov, Sutskever y Hinton. Gramática como lengua extranjera. En *Advances in Neural Information Processing Systems*, 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. Sistema de traducción de máquina neuronal de Google: Bridging the gap between human and machine translation. 2016.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li y Wei Xu. Modelos recidivos profundos con conexiones rápidas para la traducción de máquinas neuronales. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, y Jingbo Zhu. El pergamino constitutivo rápido y preciso. En las páginas 434-443. ACL, agosto de 2013.

Attention Visualizations

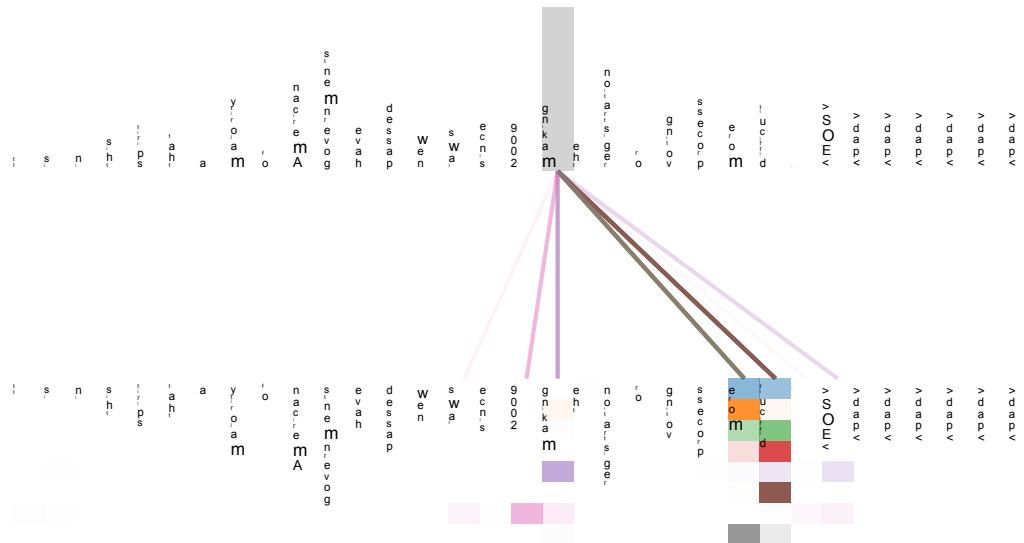


Figura 3: Ejemplo del mecanismo de atención tras las dependencias de larga distancia en la autoatención del encoder en la capa 5 de 6. Muchos de los jefes de atención atienden a una dependencia distante del verbo 'haciendo', completando la frase 'haciendo...más difícil'. Las atención aquí mostradas sólo por la palabra 'haciendo'. Diferentes colores representan diferentes cabezas. Mejor visto en color.

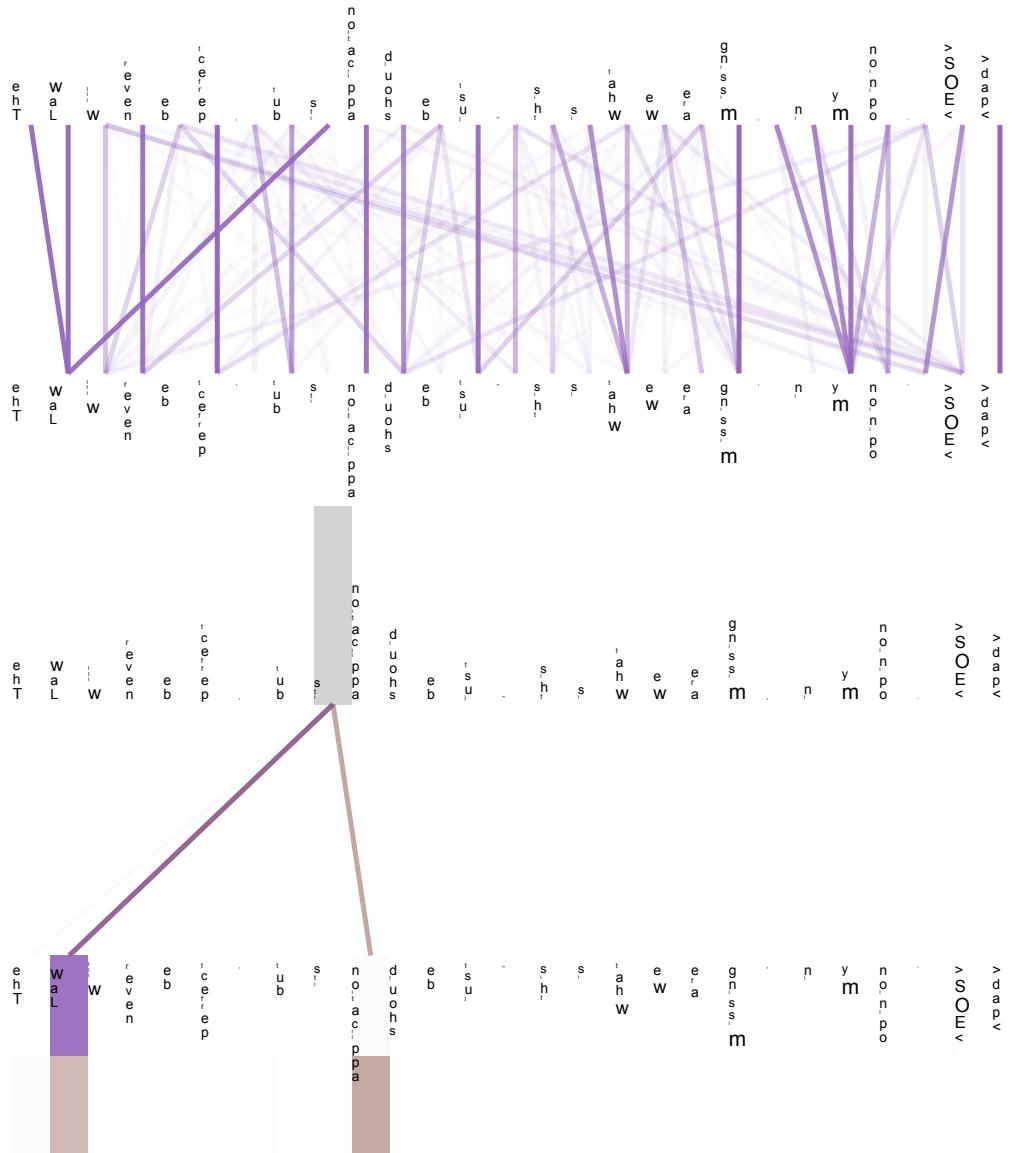


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top : Atención completa para la cabeza 5. Tema: Atenciones aisladas de sólo la palabra 'es' para los jefes 5 y 6. Tenga en cuenta que las atenciones son muy agudas para esta palabra.

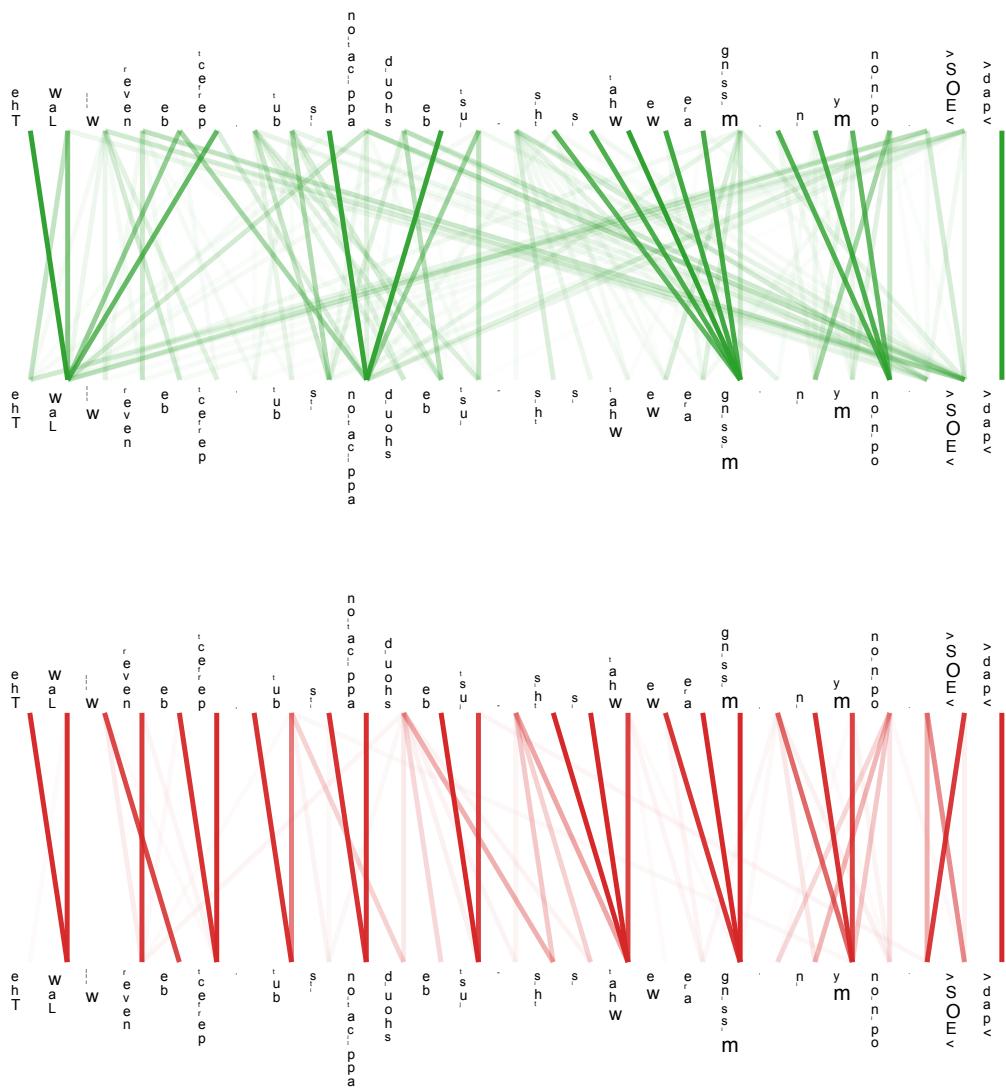


Gráfico 5: Muchos de los jefes de atención exhiben comportamientos relacionados con la estructura de la sentencia. Damos dos ejemplos más arriba, de dos cabezas diferentes de la autoatención del encoder en la capa 5 de 6. Las cabezas claramente aprendieron a realizar diferentes tareas.