

Siempre que se proporcione la atribución adecuada , Google por la presente concede permiso a reproducir las tablas y figuras de este artículo únicamente para uso periodístico o
Es un trabajo académico.

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

Káiser de Kükasz

Google Brain

lukaszkaiser@google.com

Illia Polosukhin*‡

ilia.polosukhin@gmail.com

Resumen

Los modelos de transducción de secuencia dominante se basan en complejos recurrentes o redes neuronales convolucionales que incluyen un codificador y un decodificador. Los modelos que realizan también conectan el codificador y el decodificador a través de una atención Mecanismo. Proponemos una nueva arquitectura de red simple, el transformador, basado únicamente en mecanismos de atención, prescindiendo de la recurrencia y las convoluciones completamente. Experimentos en dos tareas de traducción automática muestran estos modelos a ser superior en calidad, siendo más paralelismo y requiriendo significativamente menos tiempo para entrenar. Nuestro modelo alcanza 28.4 BLEU en el WMT 2014 Inglés-tarea de traducción al alemán, mejorando sobre los mejores resultados existentes, incluyendo ensambles, por más de 2 BLEU. En la WMT 2014 traducción Inglés-Francés tarea, nuestro modelo establece un nuevo modelo único de última generación puntuación BLEU de 41.8 después de formación de 3,5 días sobre ocho GPU, una pequeña fracción de los costes de formación de la los mejores modelos de la literatura. Demostramos que el transformador generaliza bien a otras tareas mediante su aplicación con éxito a la circunscripción inglesa grandes y limitados datos de formación.

—*El orden de inclusión en la lista es aleatorio. Jakob propuso reemplazar a las RNN con autoatención y comenzó el esfuerzo para evaluar esta idea. Ashish, con Illia, diseñó e implementó los primeros modelos de Transformador y ha participado de manera crucial en todos los aspectos de este trabajo. Noam propuso escalar la atención de los productos de punto, multi-cabeza atención y la representación de la posición libre de parámetros y se convirtió en la otra persona involucrada en casi todos Niki diseñó, implementó, afinado y evaluó innumerables variantes de modelos en nuestra base de código original y Llion también experimentó con nuevas variantes de modelos, fue responsable de nuestra base de código inicial, y inferencia y visualizaciones eficientes. Lukasz y Aidan pasaron incontables días largos diseñando varias partes de y implementar tensor2tensor, reemplazando nuestra base de código anterior, mejorando enormemente los resultados y acelerando masivamente nuestra investigación.

†Trabajo realizado mientras que en Google Brain.

‡Trabajo realizado mientras que en Google Research.

1 Introducción

Redes neuronales recidivantes, memoria a corto plazo [\[13\]](#) y redes neuronales [\[35\]](#) en particular, se han establecido firmemente como enfoques de vanguardia en el modelado secuencial y problemas de transducción tales como el modelado del lenguaje y la traducción automática [\[38\]](#). Desde entonces, los esfuerzos han seguido empujando los límites de los modelos lingüísticos recurrentes y decodificadores arquitecturas [\[24, 15\]](#).

Los modelos recurrentes suelen factorizar el cálculo a lo largo de las posiciones de símbolo de la entrada y salida secuencias. Alineando las posiciones a los pasos en el tiempo de cálculo, generan una secuencia de oculta estados, en función del estado oculto anterior y la entrada para la posición.

La naturaleza secuencial impide la paralelización dentro de los ejemplos de entrenamiento, que se vuelve crítico a largo plazo. longitudes de secuencia, ya que las restricciones de memoria limitan el loteado a través de ejemplos. mejoras significativas en la eficiencia computacional a través de trucos de factorización [\[21\]](#) computation [\[32\]](#), mientras que también mejora el rendimiento del modelo en el caso de este. Sin embargo, se mantiene la limitación de la computación secuencial.

Los mecanismos de atención se han convertido en una parte integral del modelado y transducciones, que permiten modelar las dependencias sin tener en cuenta su distancia en las secuencias de entrada o salida [\[2\]](#). En todos los casos, se utilizan conjuntamente con una red recurrente.

En este trabajo proponemos el Transformer, una arquitectura modelo evitando recurrencia y en su lugar depender enteramente de un mecanismo de atención para determinar las dependencias mundiales entre los insumos y los productos. El transformador permite una paralelización significativamente mayor y puede alcanzar un nuevo estado de la técnica en calidad de la traducción después de haber sido entrenado por tan sólo doce horas en ocho P100 GPU.

2 Antecedentes

El objetivo de reducir la computación secuencial también forma la base de la GPU Neural Extendida [\[16\]](#) y ByteNet [\[18\]](#) y ConvS2S [\[11\]](#) 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 arbitrarias de entrada o salida crece en la distancia entre posiciones, linealmente para ConvS2S y logarítmicamente para ByteNet. es más difícil aprender las dependencias entre posiciones distantes [\[12\]](#). En el transformador reducido a un número constante de operaciones, aunque a costa de una resolución efectiva reducida para promediar posiciones ponderadas por la atención, un efecto que contrarrestamos con la Atención Multi-Cabeza como descrito 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 ha sido utilizado con éxito en una variedad de tareas, incluyendo comprensión de la lectura, resumen abstracto, Envolvimiento textual y aprendizaje representaciones de frases independientes de tareas [\[4\]](#).

Las redes de memoria de extremo a extremo se basan en un mecanismo de atención recurrente en lugar de secuencia-recidiva alineada y se ha demostrado que funciona bien en la respuesta a la pregunta en lenguaje simple y tareas de modelado del lenguaje [\[34\]](#).

Sin embargo, según nuestro conocimiento, el transformador es el primer modelo de transducción que confía en enteramente en la autoatención para calcular las representaciones de su entrada y salida sin utilizar secuencia-RNN alineados o convolución. En las siguientes secciones, vamos a describir el transformador, motivar la autoatención y discutir sus ventajas sobre modelos como [\[17, 18\]](#) y [\[10\]](#).

3 Arquitectura de modelos

La mayoría de los modelos de transducción de secuencia neuronal competitivos tienen una estructura de codificador-decodificador. Aquí, el codificador mapea una secuencia de entrada de representaciones de símbolos a una secuencia de representaciones continuas. Dadas las ~~representaciones~~ de representaciones continuas, el decodificador genera entonces una salida secuencia 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

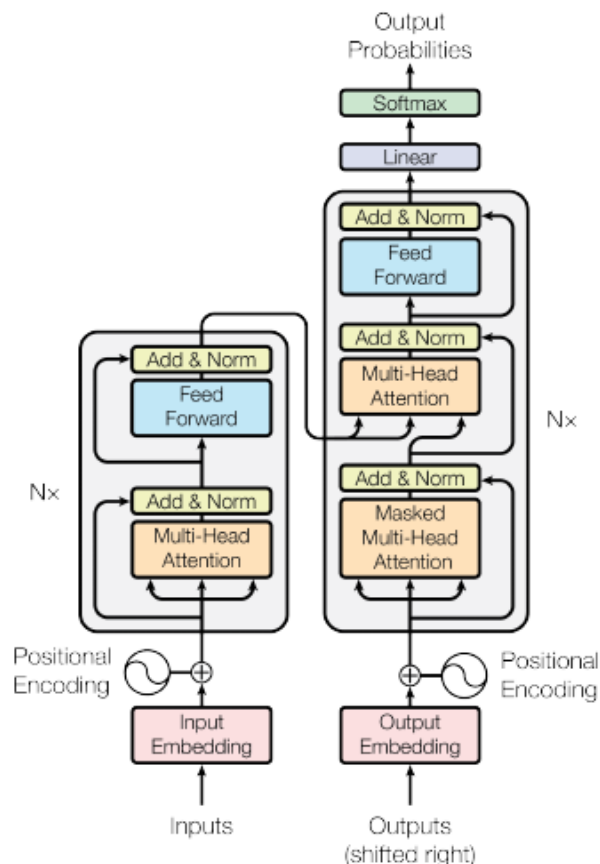


Figura 1: El transformador - arquitectura de modelos.

El transformador sigue esta arquitectura general utilizando la autoatención apilada y el punto-sabio, completamente capas conectadas tanto para el codificador como para el decodificador, mostradas 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 capas idénticas. Cada capa tiene dos sub-capas. El primero es un mecanismo de autoatención multi-cabeza, y el segundo es un simple, posición-red de avance completamente conectada. Empleamos una conexión residual [\[11\]](#) alrededor de las dos subcapas, seguidas de la normalización de capas [\[1\]](#). Es decir, la salida de cada subcapa se suma a la entrada de la capa, donde f es la función implementada por la subcapa.

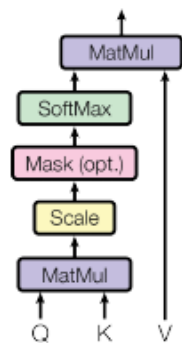
Para facilitar estas conexiones residuales, todas las subcapas en el modelo, así como la incrustación de las capas, producen salidas de dimensión d_{model} .

Decodificador El decodificador también se compone de una pila de N capas idénticas. Además de las dos sub-capas en cada capa de codificador, el decodificador inserta una tercera sub-capa, que realiza multi-cabeza de atención sobre la salida de la pila de codificador. Similar al codificador, empleamos conexiones residuales alrededor de cada una de las subcapas, seguido de la normalización de la capa. También modificamos la autoatención sub-capa en la pila decodificador para evitar que las posiciones atiendan a las posiciones posteriores. el enmascaramiento, combinado con el hecho de que las incrustaciones de salida se compensan por una posición, asegura que las predicciones para la posición t sólo puede depender de los resultados conocidos en posiciones inferiores a t .

3.2 Atención

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

Atención a los productos de puntos escalados



Atención de múltiples cabezas

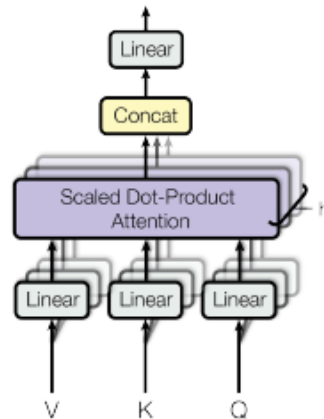


Figura 2: (izquierda) Atención a los productos de punto escalado. (derecha) La atención multicabeza consta de varios capas de atención corriendo en paralelo.

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

3.2.1 Atención a los productos de puntos escalados

Llamamos nuestra atención particular "Atención a los productos de punto escalado" (Figura [2](#)). Las consultas y claves de dimensión d_k , y valores de dimensión d_v . Calculamos los productos de punto de la consulta con todas las teclas, dividir cada uno por una función softmax para obtener los pesos en el 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 productos como:

$$A = \frac{QK^T}{\sqrt{d_k}} \quad (1)$$

Las dos funciones de atención más utilizadas son la atención aditiva [\[2\]](#) y punto-producto (o atención plicativa). La atención del producto del punto es idéntica a nuestro algoritmo, excepto para el factor de escala de $\sqrt{d_k}$. La atención aditiva calcula la función de compatibilidad usando una red de avance con una sola capa oculta. Mientras que los dos son similares en complejidad teórica, la atención del producto del punto es mucho más rápido y eficiente del espacio en la práctica, ya que se puede implementar utilizando altamente optimizado código de multiplicación de matriz.

Mientras que para los pequeños valores de d_k los dos funcionan de manera similar, la atención aditiva supera los resultados de atención del producto sin escalar para valores más grandes de d_k [\[3\]](#). Sospechamos que para grandes d_k , los productos de punto crecen grande en magnitud, empujando la función softmax en las regiones donde tiene gradientes extremadamente pequeños [\[4\]](#). Para contrarrestar este efecto, escalamos los productos de punto.

3.2.2 Atención de múltiples cabezas

En lugar de realizar una sola función de atención con d_k -claves, valores y consultas dimensionales, nos pareció beneficioso proyectar linealmente las consultas, claves y valores con diferentes aprendidos proyecciones lineales W_Q y W_K las dimensiones, respectivamente. En cada una de estas versiones proyectadas de consultas, claves y valores a continuación, realizar la función de atención en paralelo, [\[5\]](#)

— Para ilustrar por qué los productos punto se hacen grandes, asumir que los componentes son independientes al azar variables con media μ y diferencia σ . Entonces su producto punto, $\mu^2 + \sigma^2$, ha mean μ^2 y diferencia σ^2 .

valores de salida. Estos son concatenados y una vez más proyectados, resultando en los valores finales, como en la Figura 3.2.2.

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

donde

Cuando las proyecciones sean matrices de parámetros W_1, W_2, W_3 y V_1, V_2, V_3 .

En este trabajo que empleamos h capas de atención paralelas, o cabezas. Para cada uno de estos usamos W_i, V_i . Debido a la dimensión reducida de cada cabeza, el costo computacional total es similar a la de una sola cabeza de atención con plena dimensionalidad.

3.2.3 Aplicaciones de atención en nuestro modelo

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

- En capas de "atención de codificador-decodificador", las consultas provienen de la capa de decodificador anterior, y las claves de memoria y los valores provienen de la salida del codificador. Este mecanismo permite al decodificador para asistir sobre todas las posiciones en la secuencia de entrada. Este es un mecanismo típico de atención de codificador-decodificador en modelos secuencia-a-secuencia, tales como en la Figura 3.2.3.
- El codificador contiene capas de autoatención. En una capa de autoatención todas las claves, valores y las consultas vienen 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 de la codificador.
- Del mismo modo, las capas de autoatención en el decodificador permiten que cada posición en el decodificador atienda a todas las posiciones en el decodificador hasta e incluyendo esa posición. Este mecanismo permite el flujo de información en el decodificador para preservar la propiedad autorregresiva. Implementamos esto dentro de la atención de punto-producto escalado mediante el enmascaramiento (estableciendo a cero todos los valores en la entrada del softmax que corresponden a conexiones ilegales). Véase la Figura 3.2.4.

3.3 Redes de avance hacia la posición

Además de los subcapas de atención, cada una de las capas de nuestro codificador y decodificador contiene una red de avance conectado, que se aplica a cada posición de forma separada e idéntica. Esta red consiste en dos transformaciones lineales con una activación ReLU en el medio.

2)

Si bien las transformaciones lineales son las mismas en diferentes posiciones, utilizan parámetros diferentes de capa a capa. Otra forma de describir esto es como dos convoluciones con el tamaño del núcleo 1. La dimensión de entrada y salida es d , y la capa interna tiene dimensionalidad $d/2$.

3.4 Incrustaciones y Softmax

Al igual que otros modelos de transducción de secuencias, utilizamos incrustaciones aprendidas para convertir la entrada tokens y tokens de salida a vectores de dimensión d . También utilizamos el habitual transformador lineal aprendido la función softmax para convertir la salida del decodificador a las probabilidades del próximo pico predicho. En nuestro modelo, compartimos la misma matriz de peso entre las dos capas de incrustación y el pre-softmax transformación lineal, similar a la Figura 3.2.5. En las capas de incrustación, multiplicamos esos pesos

Cuadro 1: Longitudes máximas de recorrido, complejidad por capa y número mínimo de operaciones secuenciales para diferentes tipos de capas. La longitud de la secuencia, la dimensión de representación, es el núcleo y el tamaño de las convoluciones, el tamaño del barrio en autoatención restringida.

Tipo de capa	Complejidad por capa	Secuencial	Longitud máxima de la ruta
Autoatención		Operaciones	
Periódicas			
Convolutacional			
Autoatención (restricción)			

3.5 Codificación posicional

Dado que nuestro modelo no contiene recurrencia ni convolución, con el fin de que el modelo para hacer uso de la orden de la secuencia, debemos inyectar alguna información sobre la posición relativa o absoluta de la tokens en la secuencia. A este fin, añadimos "codificación posicional" a las incrustaciones de entrada en el los fondos de las pilas de codificador y decodificador. Las codificaciones posicionales tienen la misma dimensión

hay muchas opciones de codificaciones posicionales,
[\[9\].](attention.html#11)

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

donde p es la posición y d es la dimensión. Es decir, cada dimensión de la codificación posicional corresponde a un senoide. Las longitudes de onda forman una progresión geométrica de 2^d . Nosotros Elegimos esta función porque hicimos una hipótesis que permitiría que el modelo aprenda fácilmente a asistir por posiciones relativas, ya que para cualquier compensación puede ser representado como una función lineal de p .

También experimentamos con el uso de incrustaciones posicionales aprendidas [\[9\]](attention.html#11) en su lugar. Las versiones produjeron resultados casi idénticos (véase Tabla [3](attention.html#9) row (E)). Elegimos la versión [\[9\]](attention.html#9) porque puede permitir al modelo extrapolar longitudes de secuencia más largas que las encontradas durante la formación.

4 Por qué la autoatención

En esta sección se comparan varios aspectos de las capas de autoatención con los recurrentes y convolucionales comúnmente utilizadas para mapear una secuencia de longitud variable de representaciones de símbolos a otra secuencia de igual longitud, con softmax , como un oculto capa en una secuencia típica de transducción encoder o decodificador. Motivando nuestro uso de la auto-atención nosotros Considera tres desideratas.

Una es la complejidad computacional total por capa. Otra es la cantidad de cálculo que puede estar paralelizados, medidos por el número mínimo de operaciones secuenciales requeridas.

La tercera es la longitud del camino entre dependencias de largo alcance en la red. las dependencias son un reto clave en muchas tareas de transducción de secuencias. la capacidad de aprender tales dependencias es la longitud de los caminos hacia adelante y hacia atrás las señales tienen que atravesar en la red. Cuanto más cortos estos caminos entre cualquier combinación de posiciones en la entrada y secuencias de salida, más fácil es aprender dependencias de largo alcance [\[12\]](attention.html#11). Por eso tam la longitud máxima de recorrido entre dos posiciones de entrada y salida en redes diferentes tipos de capas.

Como se observa en la Tabla [6](#), una capa de autoatención conecta todas las posiciones con un número constante de operaciones ejecutadas, mientras que una capa recurrente requiere un número de operaciones que crece con la longitud de la secuencia. En términos de complejidad computacional, las capas de autoatención son más rápidas que las capas recurrentes cuando la secuencia

longitud es más pequeño que la dimensionalidad de la representación en el caso más a menudo con representaciones de frases utilizadas por modelos de última generación en traducciones automáticas, como piezas de texto [\[38\]](attention.html#12) y byte-pair [\[31\]](attention.html#12) representaciones. [\[38\]](attention.html#12) y byte-pair [\[31\]](attention.html#12) representaciones. secuencias muy largas, la auto-atención podría restringirse a considerar sólo un barrio de tamaño en la secuencia de entrada centrada alrededor de la posición de salida respectiva. Esto aumentaría el máximo longitud del camino a . Planeamos seguir investigando este enfoque en la labor futura.

Una sola capa convolucional con anchura del núcleo no conecta todos los pares de entrada y salida posiciones. Para ello se requiere una pila de capas convolucionales en el caso de granos contiguos, o en el caso de convoluciones dilatadas [\[18\]](attention.html#11), aumentando la longitud de las rutas entre cualquiera de las dos posiciones en la red. Capas convolucionales son generalmente más caros que capas recurrentes, por un factor de [\[6\]](attention.html#11), sin embargo, disminuyen la complejidad de un considerablemente, a . Incluso con , sin embargo, la complejidad de un la convolución es igual a la combinación de una capa de autoatención y una capa de avance hacia el punto, el enfoque que adoptamos en nuestro modelo.

Como beneficio secundario, la autoatención podría producir modelos más interpretables. de nuestros modelos y presentar y discutir ejemplos en el apéndice. cabezas claramente aprenden a realizar diferentes tareas, muchos parecen mostrar comportamiento relacionado con la sintaxis y estructura semántica de las oraciones.

5 Capacitación

En esta sección se describe el régimen de formación de nuestros modelos.

5.1 Datos de entrenamiento y bateo

Nos formamos en el estándar WMT 2014 conjunto de datos Inglés-Alemán que consta de unos 4,5 millones pares de frases. Las frases fueron codificadas usando codificación byte-pair [\[3\]](attention.html#10), que tiene un vocabulario objetivo de aproximadamente 37000 tokens. Para Inglés-Francés, utilizamos el WMT significativamente más gran 2014 conjunto de datos inglés-francés que consta de 36M frases y se divide tokens en un elemento de 32000 palabras vocabulario [\[38\]](attention.html#12). Los pares de sentence fueron agrupados por secuencia aproximada. lote contenía un conjunto de pares de frases que contenían aproximadamente 25000 tokens de origen y 25000 tokens objetivo.

5.2 Hardware y cronograma

Entrenamos nuestros modelos en una sola máquina con 8 GPU NVIDIA P100. los hiperparametros descritos a lo largo del papel, cada paso de entrenamiento tomó cerca de 0,4 segundos. los modelos de base para un total de 100.000 pasos o 12 horas. Para nuestros grandes modelos, (descrito en el la línea inferior de la tabla [\[3\]](attention.html#9)), step time fue de 1,0 segundos. Los grandes modelos fueron entrenados (3,5 días).

5.3 Optimizador

Utilizamos el optimizador Adam [\[20\]](attention.html#11) con Variamos el aprendizaje tasa en el curso de la formación, según la fórmula:

$$\eta_t = \eta_0 \cdot \frac{1}{1 + \alpha \cdot t} \quad (3)$$

Esto corresponde al aumento lineal de la tasa de aprendizaje para la primera $\frac{1}{\alpha}$ etapas de formación, y disminuirlo a partir de entonces proporcionalmente a la raíz cuadrada inversa del número de paso.

5.4 Regularización

Empleamos tres tipos de regularización durante el entrenamiento:

Cuadro 2: El transformador obtiene mejores puntuaciones de BLEU que los modelos anteriores de última generación en la Pruebas de inglés a alemán y de inglés a francés 2014 a una fracción del costo de la formación.

Modelo	BLEU		Costo de capacitación (FLOP)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet	23.7	39.1		
Deep-Att + PosUnk	24.6	39.9		
GNMT + RL	24.6	39.9		
ConvS2S	25.1	40.46		
MoE	32.2	40.56		
Deep-Att + PosUnk Ensemble	26.3	40.7		
GNMT + RL Ensemble	26.3	40.7		
ConvS2S Ensemble	26.3	40.7		
Transformador (modelo base)	27.3	38.1		
Transformador (grande)	28.4	41.8		

Abandono residual Aplicamos el abandono a la salida de cada subcapa, antes de que se normalice la entrada sub-capas y normalizado. Además, aplicamos la deserción a las sumas de las incrustaciones y el Codificación posicional tanto en el codificador como en las pilas decodificadoras. Para el modelo base, utilizamos una tasa de

Aislamiento de etiquetas Durante la formación, empleamos el suavizado de la etiqueta de valor de la perplejidad, ya que el modelo aprende a ser más inseguro, pero mejora la precisión y la puntuación BLEU.

6 Resultados

6.1 Traducción automática

En la tarea de traducción de inglés a alemán de la WMT 2014, el gran modelo de transformador (Transformer (big)) en la tabla 2 supera los mejores modelos notificados anteriormente (incluidos los conjuntos) de BLEU, por la que se establece una nueva puntuación de BLEU de última generación de este modelo es listada en la línea inferior de la tabla 3. El modelo de transformador (grande) entrenado para el inglés-francés utilizado supera todos los modelos y conjuntos publicados anteriormente, a una fracción del coste de formación de cualquiera de los los modelos competitivos.

En la tarea de traducción de inglés a francés de la WMT 2014, nuestro gran modelo logra una puntuación de BLEU de superar a todos los modelos individuales publicados anteriormente, a menos de coste de formación de la modelo anterior de última generación. El modelo de transformador (grande) entrenado para el inglés-francés utilizado Tasa de abandono escolar, en lugar de .

Para los modelos de base, se utilizó un único modelo obtenido con un promedio de los últimos 5 puntos de control, que Para los grandes modelos, promediamos los últimos 20 puntos de control. búsqueda de haz usado con un tamaño de haz de duración de la pena 10. Estos hiperparámetros fueron elegidos después de la experimentación en el conjunto de desarrollo. Fijamos la longitud de salida máxima durante inferencia a longitud de entrada, pero termina temprano cuando sea posible.

Tabla 2 resume nuestros resultados y compara nuestra calidad de traducción y costos de formación de las arquitecturas de la literatura. Estimamos el número de operaciones de punto flotante utilizado para entrenar un el modelo multiplicando el tiempo de entrenamiento, el número de GPU utilizados, y una estimación de Capacidad de punto flotante de una sola precisión de cada GPU.

6.2 Variaciones del modelo

Para evaluar la importancia de los diferentes componentes del transformador, hemos variado nuestro modelo base de diferentes maneras, midiendo el cambio en el rendimiento en la traducción de Inglés a Alemán en el

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 del transformador. Los valores no listados son idénticos a los de la base modelo. Todas las métricas están en el conjunto de desarrollo de la traducción de Inglés a Alemán, newstest2013. perplejidades son por pieza de la palabra, de acuerdo con nuestra codificación byte-pair, y no debe ser comparado con perplejidad por palabra.

	tren								PPL	BLEU	params	
	pasos								(dev)	(dev)		
base	6	512	2048	8	64	64	0,1	0,1	100K	4,92	25,8	65
A)									5,29	24,9		
									5,00	25,5		
									4,91	25,8		
									5,01	25,4		
B)									5,16	25,1	58	
									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
									5,77	24,6		
D)									4,95	25,5		
									4,67	25,3		
									5,47	25,7		
									4,92	25,7		
(E)	empotrado posicional en lugar de sinusoides								4,92	25,7		
grande	6	1024	4096	16				0,3	300K	4,33	26,4	213

set de desarrollo, newstest2013. Utilizamos la búsqueda de haz como se describe en la sección anterior, pero no promedio de puntos de control. Presentamos estos resultados en la Tabla [3](#).

En la Tabla [3](#) rows (A), variaremos el número de cabezas de atención y la clave de atención y mantener constante la cantidad de computación, como se describe en la sección [3.2.2](#). Mientras la atención es 0.9 BLEU peor que el mejor ajuste, la calidad también cae con demasiadas cabezas.

En la Tabla [3](#) rows (B), observamos que ~~debe la calidad del modelo~~ la elección de la clave de atención sugiere que determinar la compatibilidad no es fácil y que una compatibilidad más sofisticada función que el producto de punto puede ser beneficioso. Observamos además en las filas (C) y (D) que, como se esperaba, modelos más grandes son mejores, y la deserción es muy útil para evitar el exceso de ajuste. En la fila (E) reemplazamos nuestra Codificación posicional sinusoidal con incrustaciones posicionales aprendidas [\[9\]](#), y observamos resultados al modelo base.

6.3 El análisis de la circunscripción inglesa

Para evaluar si el transformador puede generalizar a otras tareas realizamos experimentos en inglés

Esta tarea presenta retos específicos: la producción está sujeta a fuertes

y es significativamente más larga que la entrada. Además, RNN secuencia-a-secuencia

los modelos no han sido capaces de alcanzar resultados de última generación en los regímenes de pequeños datos [\[9\]](#)

Entrenamos un transformador de 4 capas con

en la porción del Wall Street Journal (WSJ) de la

Penn Treebank [\[25\]](#), unas frases de entrenamiento de 40K. También lo entrenamos en un ent

utilizando la mayor alta confianza y BerkleyParser corpora desde con aproximadamente 17M frases

[\[37\]](#). Utilizamos un vocabulario de tokens de 16K sólo para la configuración de WSJ y un vo

para la configuración semi-supervisada.

Sólo realizamos un pequeño número de experimentos para seleccionar la deserción, tanto la atención como la residual.

(sección [5.4](#)), tasas de aprendizaje y tamaño del haz en el conjunto de desarrollo de la sección

se mantuvo sin cambios desde el modelo de traducción de base de Inglés a Alemán. Durante la inferencia,

Cuadro 4: El transformador generaliza bien el análisis de las circunscripciones inglesas (los resultados figuran en la sección 23 de WSJ)

Parser	Capacitación	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	Sólo WSJ, discriminativo	88,3
Petrov y otros (2006) [29]	Sólo WSJ, discriminativo	90,4
Zhu y otros (2013) [40]	Sólo WSJ, discriminativo	90,4
Dyer et al. (2016) [8]	Sólo WSJ, discriminativo	91,7
Transformador (4 capas)	Sólo WSJ, discriminativo	91,3
Zhu y otros (2013) [40]	Sólo WSJ, discriminativo	91,3
Huang & Harper (2009) [14]	Sólo WSJ, discriminativo	91,3
McClosky y otros (2006) [26]	Sólo WSJ, discriminativo	92,1
Vinyals & Kaiser el al. (2014) [37]	Sólo WSJ, discriminativo	92,1
Transformador (4 capas)	Sémi-supervisado	92,7
Luong y otros (2015) [23]	Sémi-supervisado	93,0
Dyer et al. (2016) [8]	Sémi-supervisado	93,3

aumentó la longitud máxima de salida a la longitud de entrada. Usamos un tamaño de haz de sólo para WSJ y la configuración semi-supervisada.

Nuestros resultados en la Tabla [\[37\]](#) muestran que a pesar de la falta de ajuste de tareas específicas sorprendentemente bien, dando mejores resultados que todos los modelos notificados anteriormente, con la excepción de la Gramática en red neural recurrente [\[8\]](#).

En contraste con los modelos de secuencia a secuencia RNN [\[37\]](#), el Transformer supera a los Parser [\[29\]](#) incluso cuando se entrena sólo en el conjunto de entrenamiento WSJ de oraciones.

7 Conclusión

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

Para las tareas de traducción, el transformador puede ser entrenado significativamente más rápido que las arquitecturas basadas en capas recurrentes o convolucionales. Tanto en WMT 2014 Inglés-Alemán y WMT 2014 Las tareas de traducción de inglés a francés, logramos un nuevo estado de la técnica. En la tarea anterior nuestra mejor modelo supera incluso a todos los conjuntos reportados anteriormente.

Estamos entusiasmados con el futuro de los modelos basados en la atención y planeamos aplicarlos a otras tareas. plan para extender el transformador a los problemas relacionados con las modalidades de entrada y salida que no sean texto y investigar mecanismos locales de atención restringida para manejar eficientemente los grandes insumos y productos como imágenes, audio y video. Hacer que la generación sea menos secuencial es otro de nuestros objetivos de investigación.

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

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

Bibliografía

- [1] Jimmy Lei Ba, Jamie Ryan Kiros y Geoffrey E Hinton. Normalización de capas. <http://arxiv.org/abs/1607.06450>, 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, y Yoshua Bengio. aprender a alinearse y traducir. , abs/1409.0473, 2014.
- [3] Denny Britz, Anna Goldie, Minh-Thang Luong y Quoc V. Le. Exploración masiva de neuronas arquitecturas de traducción automática. , abs/1703.03906, 2017.
- [4] Jianpeng Cheng, Li Dong y Mirella Lapata. Redes de memoria a corto plazo para máquinas Leyendo. <http://arxiv.org/abs/1601.06733>, 2016.

- [5] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Fethi Bougares, Holger Schwenk, y Yoshua Bengio. representaciones de frases de aprendizaje utilizando rnn codificador-decodificador para estadística traducción automática. [abs/1406.1078](http://arxiv.org/abs/1406.1078), 2014.
- [6] Francois Chollet. Xception: Aprendizaje profundo con convoluciones separables en profundidad. [abs/1610.02357](http://arxiv.org/abs/1610.02357), 2016.
- [7] Junyoung Chung, Çağlar Gülçehre, Kyunghyun Cho y Yoshua Bengio. Evaluación empírica de redes neuronales recurrentes cerradas en el modelado de secuencias. [abs/1412.3555](http://arxiv.org/abs/1412.3555), 2014.
- [8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros y Noah A. Smith. gramaticales en red. [abs/1601.05269](http://arxiv.org/abs/1601.05269), 2016.
- [9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats y Yann N. Dauphin. Secuencia de secuencia a aprendizaje de secuencia. [abs/1705.07648](http://arxiv.org/abs/1705.07648), 2017.
- [10] Alex Graves. Generando secuencias con redes neuronales recurrentes. [abs/1308.0850](http://arxiv.org/abs/1308.0850), 2013.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun. reconocimiento de la edad. [abs/1603.01525](http://arxiv.org/abs/1603.01525), páginas 770 a 778, 2016.
- [12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi y Jürgen Schmidhuber. redes recurrentes: dificultad de aprendizaje de las dependencias a largo plazo, 2001.
- [13] Sepp Hochreiter y Jürgen Schmidhuber. Memoria larga a corto plazo. [abs/1908.08755](http://arxiv.org/abs/1908.08755), 9(8):1735–1780, 1997.
- [14] Zhongqiang Huang y Mary Harper. Gramáticas PCFG autoentrenadas con anotaciones latentes a través de los idiomas. [abs/0908.3480](http://arxiv.org/abs/0908.3480), págs. 832 a 841; ACL, agosto de 2009.
- [15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer y Yonghui Wu. Explorando los límites de la modelización lingüística. [abs/1602.02410](http://arxiv.org/abs/1602.02410), 2016.
- [16] Káiser y Samy Bengio. ¿Puede la memoria activa reemplazar la atención? [abs/1603.04467](http://arxiv.org/abs/1603.04467), 2016.
- [17] Káiser e Ilya Sutskever. Las GPUs neuronales aprenden algoritmos. [abs/1603.04467](http://arxiv.org/abs/1603.04467), 2016.
- [18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves y Koray Kavukcuoglu. Traducción automática neural en tiempo lineal. [abs/1610.05098](http://arxiv.org/abs/1610.05098), 2017.
- [19] Yoon Kim, Carl Denton, Luong Hoang y Alexander M. Rush. Redes de atención estructuradas. In [abs/1708.03782](http://arxiv.org/abs/1708.03782), 2017.
- [20] Diederik Kingma y Jimmy Ba. Adam: Un método para la optimización estocástica. [abs/1412.0441](http://arxiv.org/abs/1412.0441), 2015.
- [21] Oleksii Kuchaiev y Boris Ginsburg. Trucos de factorización para redes LSTM. [abs/1703.10722](http://arxiv.org/abs/1703.10722), 2017.
- [22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, y Yoshua Bengio. Una frase estructurada auto-atención incrustada. [abs/1703.03130](http://arxiv.org/abs/1703.03130), 2017.
- [23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals y Lukasz Kaiser. secuencia a secuencia de aprendizaje. [abs/1511.06114](http://arxiv.org/abs/1511.06114), 2015.
- [24] Minh-Thang Luong, Hieu Pham y Christopher D Manning. traducción automática neuronal basada. [abs/1508.04025](http://arxiv.org/abs/1508.04025), 2015.

- [25] Mitchell P Marcus, Mary Ann Marcinkiewicz y Beatrice Santorini. corpus de inglés: El banco del árbol del penn. , 19(2):313–330, 1993.
- [26] David McClosky, Eugene Charniak, y Mark Johnson. Auto-entrenamiento eficaz para el análisis. ACL, junio de 2006.
- [27] Ankur Parikh, Oscar Täckström, Dipanjan Das y Jakob Uszkoreit. Una atención descompuesta modelo. En , 2016.
- [28] Romain Paulus, Caiming Xiong, y Richard Socher. Un modelo profundamente reforzado para la abstracción resumen. [\[redacted\] <a href="http://arxiv.org/abs/1705.04304", 2017.](http://arxiv.org/abs/1705.04304)
- [29] Slav Petrov, Leon Barrett, Romain Thibaux y Dan Klein. y anotación de árbol interpretable. En , págs. 433 a 440. ACL, julio 2006.
- [30] Ofir Press y Lior Wolf. Usando la inserción de salida para mejorar los modelos lingüísticos. [\[redacted\] <a href="http://arxiv.org/abs/1608.05859", 2016.](http://arxiv.org/abs/1608.05859)
- [31] Rico Sennrich, Barry Haddow y Alexandra Birch. Traducción automática neural de palabras raras con unidades de subpalabra. [\[redacted\] <a href="http://arxiv.org/abs/1508.07909", 2015.](http://arxiv.org/abs/1508.07909)
- [32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarsz, Andy Davis, Quoc Le, Geoffrey Hinton, y Jeff Dean. Red neuronal escandalosamente grande: La mezcla escasamente cerrada de expertos capa. [\[redacted\] <a href="http://arxiv.org/abs/1701.06538", 2017.](http://arxiv.org/abs/1701.06538)
- [33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever y Ruslan Salakhutdinov. Abandono: una forma sencilla de evitar que las redes neuronales se ajusten demasiado. , 15(1):1929-1958, 2014.
- [34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston y Rob Fergus. Memoria de extremo a extremo En C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama y R. Garnett, editores, , págs. 2440 a 2448. Curran Associates, Inc., 2015.
- [35] Ilya Sutskever, Oriol Vinyals y Quoc VV Le. Secuencia para el aprendizaje de secuencias con neural en las redes. , págs. 3104 a 3112, 2014.
- [36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens y Zbigniew Wojna. Repensar la arquitectura inicial para la visión computarizada. , abs/1512.00567, 2015.
- [37] Vinyals & Kaiser, Koo, Petrov, Sutskever, y Hinton. Gramática como lengua extranjera. En , 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. Máquina neuronal de Google sistema de traducción: Superando la brecha entre la traducción humana y la traducción automática. [\[redacted\] <a href="http://arxiv.org/abs/1609.08144", 2016.](http://arxiv.org/abs/1609.08144)
- [39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, y Wei Xu. conexiones rápidas para la traducción automática neural. , abs/1606.04199, 2016.
- [40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang y Jingbo Zhu. Rápido y preciso cambio-reducir el análisis constituyente. In , págs. 434 a 443. ACL, agosto de 2013.

Visualizaciones de atención

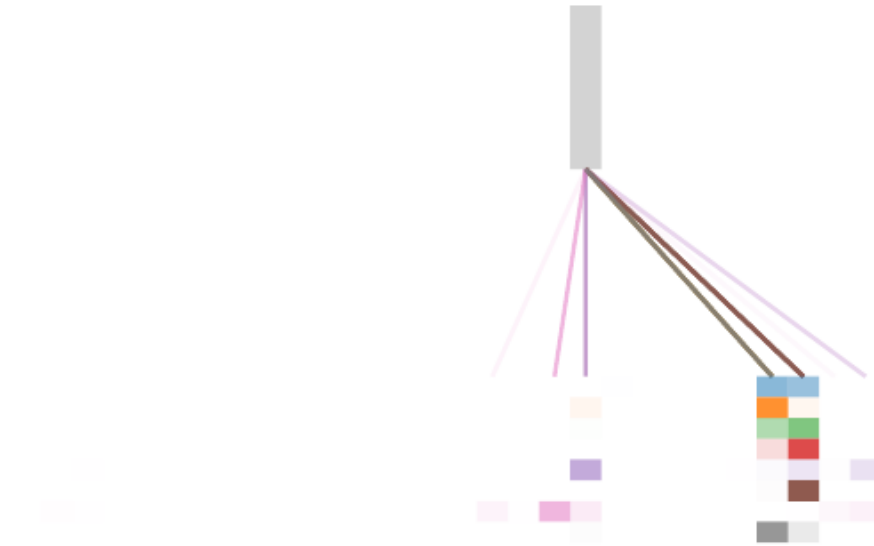


Figura 3: Ejemplo del mecanismo de atención que sigue a las dependencias de larga distancia encoder auto-atención en la capa 5 de 6. Muchos de los jefes de la atención atienden a una dependencia distante de el verbo 'hacer', completando la frase 'hacer...más difícil'.
la palabra 'hacer'. Diferentes colores representan diferentes cabezas. Mejor visto en color.

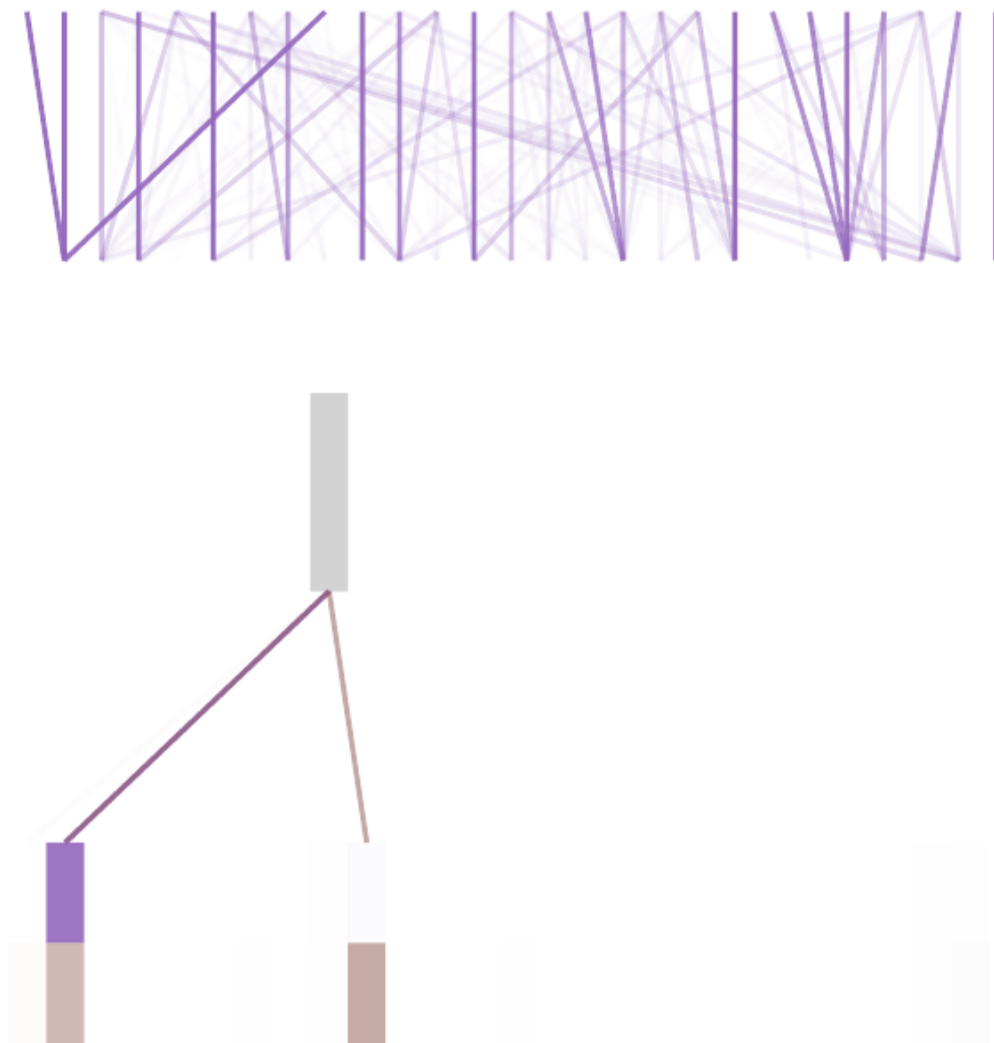


Figura 4: Dos cabezas de atención, también en la capa 5 de 6, aparentemente involucradas en la resolución de anafora. Atenciones completas para la cabeza 5. Abajo: Atenciones aisladas de sólo la palabra 'su' para las cabezas de atención 5 6. Tenga en cuenta que las atenciones son muy agudas para esta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

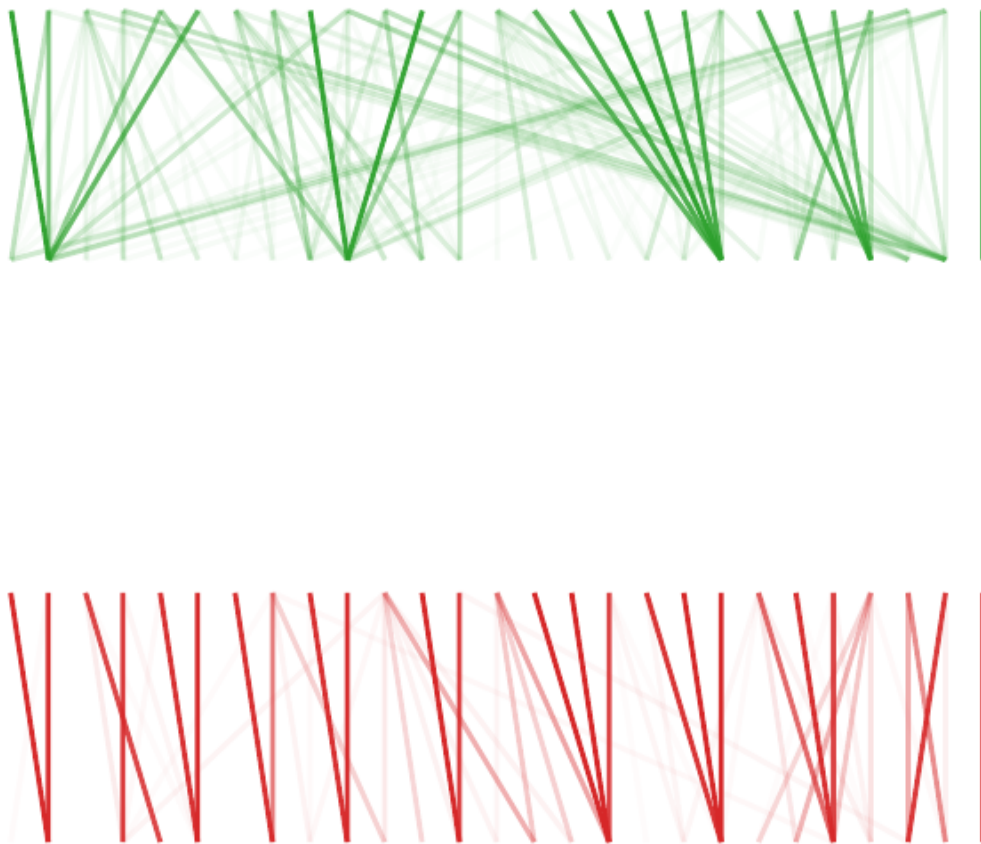


Figura 5: Muchos de los jefes de atención muestran un comportamiento que parece estar relacionado con la estructura de la frase. Damos dos ejemplos de este tipo arriba, de dos cabezas diferentes de la auto-atención del codificador en la capa 5 de 6. Las cabezas claramente aprendieron a realizar diferentes tareas.