Atenção é tudo que você precisa

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Pesquisa do Google
nikip@google.com

Jakob Uszkoreit*
Pesquisa do Google
usz@google.com

Llion Jones*
Pesquisa do Google
llion@google.com

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

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin*‡
illia.polosukhin@gmail.com

Abstrato

Os modelos de transdução de sequência dominante são baseados em redes neurais recorrentes ou convolucionais complexas que incluem um codificador e um decodificador. Os modelos de melhor desempenho também conectam o codificador e o decodificador por meio de um mecanismo de atenção. Propomos uma nova arquitetura de rede simples, o Transformer, baseada apenas em mecanismos de atenção, dispensando totalmente a recorrência e as convoluções. Experimentos em duas tarefas de tradução automática mostram que esses modelos são superiores em qualidade, sendo mais paralelizáveis e exigindo significativamente menos tempo para treinar. Nosso modelo atinge 28,4 BLEU na tarefa de tradução de inglês para alemão do WMT 2014, melhorando os melhores resultados existentes, incluindo conjuntos, em mais de 2 BLEU. Na tarefa de tradução de inglês para francês do WMT 2014, nosso modelo estabelece uma nova pontuação BLEU de modelo único de última geração de 41. 8 após treinamento de 3,5 dias em oito GPUs, uma pequena fração dos custos de treinamento dos melhores modelos da literatura. Mostramos que o Transformer generaliza bem para outras tarefas, aplicando-o com sucesso à análise de constituintes ingleses com dados de treinamento grandes e limitados.

*Contribuição igual. A ordem de listagem é aleatória. Jakob propôs substituir RNNs por autoatenção e iniciou o esforço para avaliar essa ideia. Ashish, com Illia, projetou e implementou os primeiros modelos Transformer e esteve crucialmente envolvido em todos os aspectos deste trabalho. Noam propôs a atenção de produto escalar, a atenção de várias cabeças e a representação de posição livre de parâmetros e tornou-se a outra pessoa envolvida em quase todos os detalhes. Niki projetou, implementou, ajustou e avaliou inúmeras variantes de modelo em nossa base de código original e tensor2tensor. Llion também experimentou novas variantes de modelo, foi responsável por nossa base de código inicial e por inferências e visualizações eficientes. Lukasz e Aidan passaram incontáveis longos dias projetando várias partes e implementando o tensor2tensor, substituindo nossa base de código anterior,

†Trabalho realizado no Google Brain.
‡Trabalho realizado no Google Research.

31ª Conferência sobre Sistemas de Processamento de Informação Neural (NIPS 2017), Long Beach, CA, EUA.

1. Introdução

Redes neurais recorrentes, memória de longo prazo [13] e redes neurais recorrentes [7] em particular, foram firmemente estabelecidas como abordagens de última geração em modelagem de sequência e problemas de transdução, como modelagem de linguagem e tradução automática [35, 2 , 5]. Numerosos esforços desde então continuaram a empurrar os limites dos modelos de linguagem recorrentes e arquiteturas codificador-decodificador [38, 24, 15].

Os modelos recorrentes normalmente fatoram a computação ao longo das posições dos símbolos das sequências de entrada e saída. Alinhando as posições a passos em tempo de computação, eles geram uma sequência de estados ocultos h_t, em função do estado oculto anterior h_{t-1}e a entrada para a posição t. Essa natureza inerentemente sequencial impede a paralelização nos exemplos de treinamento, o que se torna crítico em comprimentos de sequência mais longos, pois as restrições de memória limitam o lote entre os exemplos. Trabalhos recentes alcançaram melhorias significativas na eficiência computacional por meio de truques de fatoração [21] e computação condicional [32], além de melhorar o desempenho do modelo no caso do último. A restrição fundamental da computação sequencial, no entanto, permanece.

Mecanismos de atenção tornaram-se parte integrante da modelagem de sequência e modelos de transdução em várias tarefas, permitindo a modelagem de dependências sem levar em consideração sua distância nas sequências de entrada ou saída [2, 19]. Em quase todos os casos [27], no entanto, tais mecanismos de atenção são usados em conjunto com uma rede recorrente.

Neste trabalho, propomos o Transformer, uma arquitetura de modelo que evita a recorrência e, em vez disso, depende inteiramente de um mecanismo de atenção para desenhar dependências globais entre entrada e saída. O Transformer permite significativamente mais paralelização e pode atingir um novo estado da arte em qualidade de tradução depois de ser treinado por apenas doze horas em oito GPUs P100.

2. Antecedentes

O objetivo de reduzir a computação sequencial também forma a base da GPU neural estendida [16], ByteNet [18] e ConvS2S [9], todos os quais usam redes neurais convolucionais como bloco de construção básico, computando representações ocultas em paralelo para todas as entradas e posições de saída. Nesses modelos, o número de operações necessárias para relacionar os sinais de duas posições arbitrárias de entrada ou saída cresce na distância entre as posições, linearmente para ConvS2S e logaritmicamente para ByteNet. Isso torna mais difícil aprender dependências entre posições distantes [12]. No Transformer, isso é reduzido a um número constante de operações, embora ao custo de uma resolução efetiva reduzida devido à média das posições de atenção ponderada, um efeito que neutralizamos com a atenção multicabeça, conforme descrito na seção 3.2.

A auto-atenção, às vezes chamada de intra-atenção, é um mecanismo de atenção que relaciona diferentes posições de uma única sequência para calcular uma representação da sequência. A auto-atenção tem sido usada com sucesso em uma variedade de tarefas, incluindo compreensão de leitura, resumo abstrativo, vinculação textual e representações de sentenças independentes da tarefa de aprendizagem [4, 27, 28, 22].

As redes de memória fim-a-fim são baseadas em um mecanismo de atenção recorrente em vez de recorrência alinhada por sequência e demonstraram ter um bom desempenho em tarefas de resposta a perguntas de linguagem simples e tarefas de modelagem de linguagem [34].

Até onde sabemos, no entanto, o Transformer é o primeiro modelo de transdução que depende inteiramente da autoatenção para calcular representações de sua entrada e saída sem usar RNNs ou convolução alinhados à sequência. Nas seções a seguir, descreveremos o Transformer, motivaremos a autoatenção e discutiremos suas vantagens sobre modelos como [17, 18] e [9].

3 Modelo de Arquitetura

A maioria dos modelos competitivos de transdução de sequência neural tem uma estrutura codificador-decodificador [5, 2, 35]. Aqui, o codificador mapeia uma sequência de entrada de representações de símbolos (x_1 , ...,x_n)a uma sequência de representações contínuasz = (z_1, ..., z_n). Dadoz,o decodificador então gera uma sequência de saída (y_1, ..., $você_m$)de símbolos um elemento de cada vez. A cada passo o modelo é auto-regressivo [10], consumindo os símbolos gerados anteriormente como entrada adicional ao gerar o próximo.

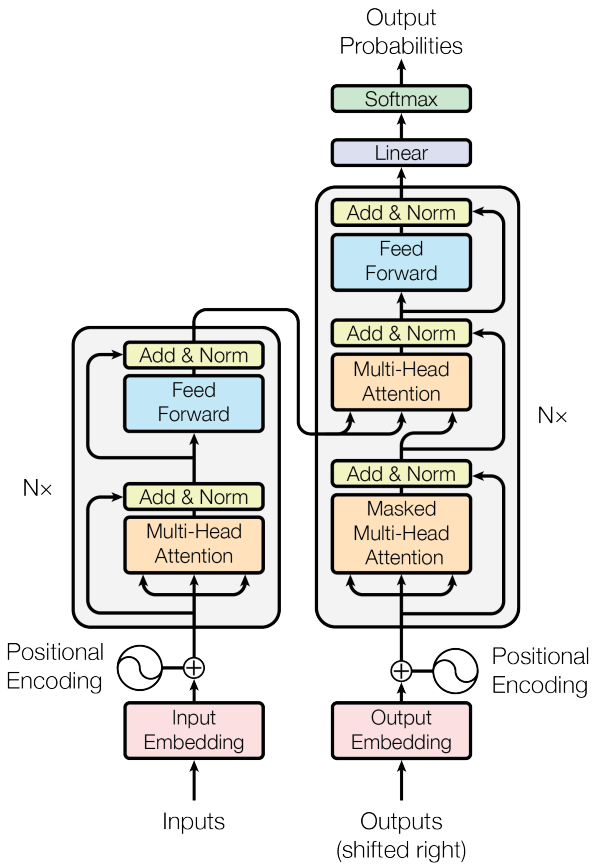


Figura 1: O Transformer - arquitetura do modelo.

O Transformer segue essa arquitetura geral usando autoatenção empilhada e camadas totalmente conectadas por pontos para o codificador e o decodificador, mostrados nas metades esquerda e direita da Figura 1, respectivamente.

3.1 Pilhas de codificador e decodificador

Codificador:O codificador é composto por uma pilha de$N$=6camadas idênticas. Cada camada tem duas subcamadas. O primeiro é um mecanismo de auto-atenção de múltiplas cabeças, e o segundo é uma rede feed-forward simples e totalmente conectada posicionalmente. Empregamos uma conexão residual [11] em torno de cada uma das duas subcamadas, seguida de normalização de camada [1]. Ou seja, a saída de cada subcamada é LayerNorm(x+Subcamada(x)),ondeSubcamada(x)é a função implementada pela própria subcamada. Para facilitar essas conexões residuais, todas as subcamadas do modelo, bem como as camadas de incorporação, produzem saídas de dimensãod_{modelo}=512.

Decodificador:O decodificador também é composto por uma pilha de$N$=6camadas idênticas. Além das duas subcamadas em cada camada do codificador, o decodificador insere uma terceira subcamada, que executa atenção multicabeçal na saída da pilha do codificador. Semelhante ao codificador, empregamos conexões residuais em torno de cada uma das subcamadas, seguidas pela normalização da camada. Também modificamos a subcamada de auto-atenção na pilha do decodificador para evitar que as posições atendam às posições subsequentes. Esse mascaramento, combinado com o fato de que as incorporações de saída são compensadas por uma posição, garante que as previsões de posiçãoeupode depender apenas das saídas conhecidas em posições menores queeu.

3.2 Atenção

Uma função de atenção pode ser descrita como mapear uma consulta e um conjunto de pares chave-valor para uma saída, onde a consulta, as chaves, os valores e a saída são todos vetores. A saída é calculada como uma soma ponderada

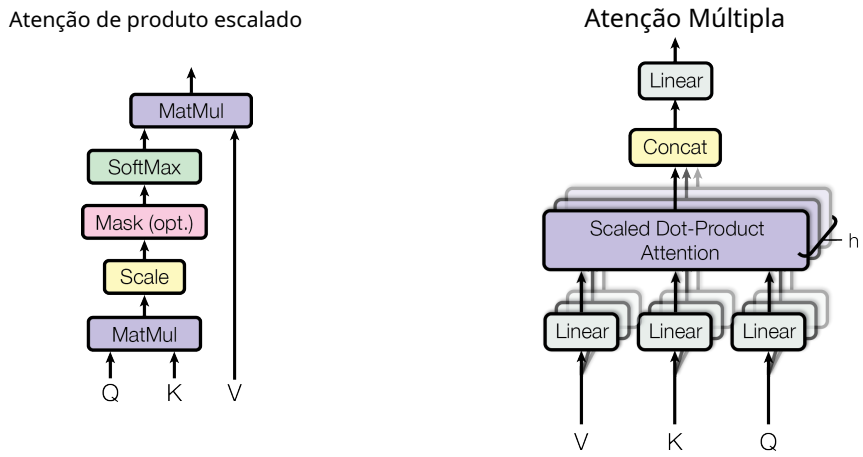


Figura 2: (esquerda) Atenção escalado de produto escalar. (à direita) A atenção multicabeçal consiste em várias camadas de atenção executadas em paralelo.

dos valores, onde o peso atribuído a cada valor é calculado por uma função de compatibilidade da consulta com a chave correspondente.

3.2.1 Atenção escalado de produto escalar

Chamamos nossa atenção especial "Atenção escalado do produto escalar" (Figura 2). A entrada consiste em consultas e chaves de dimensão d_k, a $\sqrt{}$ nd valores de dimensão d_v. Calculamos os produtos escalares da consulta com todas as chaves, divida cada uma por d_k, e aplique uma função softmax para obter os pesos nos valores.

Na prática, calculamos a função de atenção em um conjunto de consultas simultaneamente, agrupadas em uma matriz Q. As chaves e valores também são agrupados em matrizes k e V. Calculamos a matriz de saídas como:

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

As duas funções de atenção mais usadas são a atenção aditiva [2] e a atenção de produto escalar (multiplicativa). A atenção do produto escalar é idêntica ao nosso algoritmo, exceto pelo fator de escala de $\frac{1}{\sqrt{d_k}}$. A atenção aditiva calcula a função de compatibilidade usando uma rede feed-forward com uma única camada oculta. Embora os dois sejam semelhantes em complexidade teórica, a atenção ao produto escalar é muito mais rápida e eficiente em termos de espaço na prática, pois pode ser implementada usando um código de multiplicação de matrizes altamente otimizado.

Enquanto para pequenos valores de d_k os dois mecanismos funcionam de maneira semelhante, a atenção aditiva supera a atenção do produto escalar sem dimensionar para valores maiores de d_k [3]. Suspeitamos que para grandes valores de d_k, os produtos escalares crescem em magnitude, empurrando a função softmax para regiões onde ela possui gradientes extremamente pequenos[4]. Para neutralizar esse efeito, escalamos os produtos escalares por $\frac{1}{\sqrt{d_k}}$.

3.2.2 Atenção Múltipla

Em vez de realizar uma única função de atenção com d_{modelo} chaves, valores e consultas tridimensionais, achamos útil projetar linearmente as consultas, chaves e valores h vezes com diferentes projeções lineares aprendidas para d_k, d_k e d_v dimensões, respectivamente. Em cada uma dessas versões projetadas de consultas, chaves e valores, executamos a função de atenção em paralelo, produzindo d_v-dimensional

[4] Para ilustrar por que os produtos escalares ficam grandes, suponha que os componentes de q e k são aleatórios independentes variáveis com média 0 e variância 1. Em seguida, seu produto escalar, $q \cdot k = \sum_{eu=1}^{d_k} q_{eu} k_{eu}$, significa 0 e variância d_k.

valores de saída. Estes são concatenados e novamente projetados, resultando nos valores finais, conforme ilustrado na Figura 2.

A atenção multicabeça permite que o modelo atenda conjuntamente a informações de diferentes subespaços de representação em diferentes posições. Com uma única cabeça de atenção, a média inibe isso.

$$\text{Cabeçalho múltiplo}(Q, K, V) = \text{Concat}(\text{cabeça}_1, \ldots, \text{cabeça}_h)C^O$$
$$\text{ondecabeça}_{eu} = \text{Atenção}(QW^Q_{eu}, KW^k_{eu}, VW^V_{eu})$$

Onde as projeções são matrizes de parâmetros $C^Q_{eu} \in \mathbb{R}^{d_{\text{modelo}} \times d_k}$, $C^k_{eu} \in \mathbb{R}^{d_{\text{modelo}} \times d_k}$, $C^V_{eu} \in \mathbb{R}^{d_{\text{modelo}} \times d_v}$ e $C^O \in \mathbb{R}^{hd_v \times d_{\text{modelo}}}$.

Neste trabalho empregamos h=8camadas paralelas de atenção, ou cabeças. Para cada um deles usamos $d_k = d_v = d_{\text{modelo}}/h$=64.Devido à dimensão reduzida de cada cabeça, o custo computacional total é semelhante ao da atenção de cabeça única com dimensionalidade total.

3.2.3 Aplicações da Atenção em nosso Modelo

O Transformer usa a atenção de várias cabeças de três maneiras diferentes:

- Nas camadas "atenção do codificador-decodificador", as consultas vêm da camada do decodificador anterior, e as chaves de memória e os valores vêm da saída do codificador. Isso permite que cada posição no decodificador atenda a todas as posições na sequência de entrada. Isso imita os mecanismos típicos de atenção do codificador-decodificador em modelos sequência a sequência, como [38, 2, 9].
- O codificador contém camadas de auto-atenção. Em uma camada de auto-atenção todas as chaves, valores e consultas vêm do mesmo lugar, neste caso, a saída da camada anterior no codificador. Cada posição no codificador pode atender a todas as posições na camada anterior do codificador.
- Da mesma forma, as camadas de auto-atenção no decodificador permitem que cada posição no decodificador atenda a todas as posições no decodificador até e incluindo essa posição. Precisamos impedir o fluxo de informações para a esquerda no decodificador para preservar a propriedade auto-regressiva. Implementamos isso dentro da atenção escalada do produto escalar mascarando (configurando para $-\infty$) todos os valores na entrada do softmax que correspondem a conexões ilegais. Veja a Figura 2.

3.3 Redes de feed-forward por posição

Além das subcamadas de atenção, cada uma das camadas em nosso codificador e decodificador contém uma rede de feed-forward totalmente conectada, que é aplicada a cada posição separadamente e de forma idêntica. Isso consiste em duas transformações lineares com uma ativação ReLU entre elas.

$$\text{FFN}(x) = \text{máx}(0, xW_1 + b_1)C_2 + b_2 \quad (2)$$

Embora as transformações lineares sejam as mesmas em diferentes posições, elas usam parâmetros diferentes de camada para camada. Outra maneira de descrever isso é como duas convoluções com tamanho de kernel 1. A dimensionalidade de entrada e saída é d_{modelo}=512,e a camada interna tem dimensionalidade d_{ff}=2048.

3.4 Incorporações e Softmax

Da mesma forma que outros modelos de transdução de sequência, usamos embeddings aprendidos para converter os tokens de entrada e os tokens de saída em vetores de dimensão d_{modelo}. Também usamos a transformação linear aprendida usual e a função softmax para converter a saída do decodificador em probabilidades de próximo token previstas. Em nosso modelo, compartilhamos a mesma matriz de peso entre as duas camadas de incorporação e a transformação linear, semelhante a [30]. Nas camadas de incorporação, multiplicamos esses pesos por $\sqrt{d_{\text{modelo}}}$.

Tabela 1: Comprimentos máximos de caminho, complexidade por camada e número mínimo de operações sequenciais para diferentes tipos de camada. n é o comprimento da sequência, d é a dimensão da representação, k é o tamanho do kernel das convoluções e r o tamanho da vizinhança em autoatenção restrita.

Tipo de Camada	Complexidade por Camada	Sequencial Operações	Comprimento Máximo do Caminho
Autoatenção	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recorrente	$O(n \cdot d^2)$	$O(n)$	$O(n)$
convolucional	$O(k \cdot n \cdot d^2)$ $O($	$O(1)$	$O(registro_k(n))$
Autoatenção (restrita)	$r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Codificação Posicional

Como nosso modelo não contém recorrência e nem convolução, para que o modelo faça uso da ordem da sequência, devemos injetar algumas informações sobre a posição relativa ou absoluta dos tokens na sequência. Para esse fim, adicionamos "codificações posicionais" às incorporações de entrada na parte inferior das pilhas do codificador e do decodificador. As codificações posicionais têm a mesma dimensão d_{modelo} como os embeddings, para que os dois possam ser somados. Existem muitas opções de codificações posicionais, aprendidas e fixas [9].

Neste trabalho, utilizamos as funções seno e cosseno de diferentes frequências:

$$EDUCAÇAO\ FISICA_{(pos,2eu)} = pecado(pos/10000^{2eu\ ia_{modelo}})$$

$$EDUCAÇÃO\ FISICA_{(pos,2eu+1)} = porque(pos/10000^{2eu\ ia_{modelo}})$$

onde pos é a posição e eu é a dimensão. Ou seja, cada dimensão da codificação posicional corresponde a uma senóide. Os comprimentos de onda formam uma progressão geométrica de 2π para $10000 \cdot 2\pi$. Escolhemos esta função porque imaginamos que ela permitiria que o modelo aprendesse facilmente a atender por posições relativas, pois para qualquer deslocamento fixo k, $EDUCAÇAO\ FISICA_{pos+k}$ pode ser representado como uma função linear de $EDUCAÇAO\ FISICA_{pos}$.

Em vez disso, também experimentamos o uso de incorporações posicionais aprendidas [9] e descobrimos que as duas versões produziram resultados quase idênticos (consulte a tabela 3, linha (E)). Escolhemos a versão senoidal porque pode permitir que o modelo extrapole para comprimentos de sequência maiores do que os encontrados durante o treinamento.

4 Por que auto-atenção

Nesta seção, comparamos vários aspectos das camadas de auto-atenção com as camadas recorrentes e convolucionais comumente usadas para mapear uma sequência de comprimento variável de representações de símbolos $(x_1, ..., x_n)$ para outra sequência de igual comprimento $(z_1, ..., z_n)$, com $x_{eu}, z_{eu} \in \mathbb{R}^d$, como uma camada oculta em um codificador ou decodificador de transdução de sequência típico. Para motivar nosso uso da autoatenção, consideramos três desideratos.

Uma delas é a complexidade computacional total por camada. Outra é a quantidade de computação que pode ser paralelizada, medida pelo número mínimo de operações sequenciais necessárias.

O terceiro é o comprimento do caminho entre dependências de longo alcance na rede. Aprender dependências de longo alcance é um desafio chave em muitas tarefas de transdução de sequência. Um fator-chave que afeta a capacidade de aprender essas dependências é o comprimento dos caminhos que os sinais de avanço e retorno devem percorrer na rede. Quanto mais curtos esses caminhos entre qualquer combinação de posições nas sequências de entrada e saída, mais fácil é aprender dependências de longo alcance [12]. Portanto, também comparamos o comprimento máximo do caminho entre quaisquer duas posições de entrada e saída em redes compostas pelos diferentes tipos de camadas.

Conforme observado na Tabela 1, uma camada de auto-atenção conecta todas as posições com um número constante de operações executadas sequencialmente, enquanto uma camada recorrente requer $O(n)$ operações sequenciais. Em termos de complexidade computacional, as camadas de auto-atenção são mais rápidas que as camadas recorrentes quando a sequência

comprimento n é menor que a dimensionalidade de representação d, que é o caso mais frequente com representações de sentenças usadas por modelos de última geração em traduções automáticas, como representações de palavras [38] e pares de bytes [31]. Para melhorar o desempenho computacional em tarefas que envolvem sequências muito longas, a autoatenção pode ser restrita a considerar apenas uma vizinhança de tamanho r na sequência de entrada centrada em torno da respectiva posição de saída. Isso aumentaria o comprimento máximo do caminho para $O(n/r)$. Pretendemos aprofundar esta abordagem em trabalhos futuros.

Uma única camada convolucional com largura de kernel $k < n$ não conecta todos os pares de posições de entrada e saída. Fazer isso requer uma pilha de $O(n/k)$ camadas convolucionais no caso de núcleos contíguos, ou $O(registro_k(n))$ no caso de convoluções dilatadas [18], aumentando o comprimento dos caminhos mais longos entre quaisquer duas posições na rede. As camadas convolucionais são geralmente mais caras do que as camadas recorrentes, por um fator de k. Convoluções separáveis [6], no entanto, diminuem consideravelmente a complexidade, para $O(k \cdot n \cdot d + n \cdot d^2)$. Mesmo com $k = n$, no entanto, a complexidade de uma convolução separável é igual à combinação de uma camada de auto-atenção e uma camada de feed-forward pontual, a abordagem que adotamos em nosso modelo.

Como benefício colateral, a autoatenção pode produzir modelos mais interpretáveis. Inspecionamos as distribuições de atenção de nossos modelos e apresentamos e discutimos exemplos no apêndice. As cabeças de atenção individuais não apenas aprendem claramente a realizar tarefas diferentes, mas muitas parecem exibir comportamentos relacionados à estrutura sintática e semântica das sentenças.

5 Treinamento

Esta seção descreve o regime de treinamento para nossos modelos.

5.1 Dados de treinamento e lotes

Treinamos com o conjunto de dados inglês-alemão padrão WMT 2014, que consiste em cerca de 4,5 milhões de pares de sentenças. As sentenças foram codificadas usando a codificação de pares de bytes [3], que possui um vocabulário fonte-alvo compartilhado de cerca de 37.000 tokens. Para inglês-francês, usamos o conjunto de dados inglês-francês WMT 2014 significativamente maior, que consiste em 36 milhões de sentenças e tokens divididos em um vocabulário de 32.000 palavras [38]. Pares de sentenças foram agrupados por comprimento de sequência aproximado. Cada lote de treinamento continha um conjunto de pares de sentenças contendo aproximadamente 25.000 tokens de origem e 25.000 tokens de destino.

5.2 Hardware e Cronograma

Treinamos nossos modelos em uma máquina com 8 GPUs NVIDIA P100. Para nossos modelos básicos usando os hiperparâmetros descritos ao longo do artigo, cada etapa de treinamento levou cerca de 0,4 segundos. Treinamos os modelos básicos para um total de 100.000 passos ou 12 horas. Para nossos modelos grandes (descritos na última linha da tabela 3), o tempo de passo foi de 1,0 segundos. Os grandes modelos foram treinados por 300.000 passos (3,5 dias).

5.3 Otimizador

Usamos o otimizador Adam [20] com $\beta_1 = 0.9$, $\beta_2 = 0.98$ e $\epsilon = 10^{-9}$. Variamos a taxa de aprendizado ao longo do treinamento, de acordo com a fórmula:

$$lrate = d_{modelo}^{-0.5} \cdot \min(etapa\_num^{-0.5}, etapa\_num \cdot aquecimento\_passos^{-1.5}) \quad (3)$$

Isso corresponde a aumentar a taxa de aprendizado linearmente para o primeiro $aquecimento\_passos$ etapas de treinamento e diminuindo-a posteriormente proporcionalmente à raiz quadrada inversa do número da etapa. Nós costumavamos $aquecimento\_passos = 4000$.

5.4 Regularização

Empregamos três tipos de regularização durante o treinamento:

Tabela 2: O Transformer alcança melhores pontuações BLEU do que os modelos anteriores de última geração nos testes newstest2014 de inglês para alemão e de inglês para francês por uma fração do custo do treinamento.

Modelo	AZUL		Custo de treinamento (FLOPs)	
	EN-DE	EN-FR	EN-DE	PT-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] ConvS2S	26,30	41,16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformador (modelo básico)	27,3	38,1	$3.3 \cdot 10^{18}$	
Transformador (grande)	28,4	41,8	$2.3 \cdot 10^{19}$	

Desistência residualAplicamos dropout [33] à saída de cada subcamada, antes de ser adicionada à entrada da subcamada e normalizada. Além disso, aplicamos dropout às somas dos embeddings e das codificações posicionais nas pilhas do codificador e do decodificador. Para o modelo base, usamos uma taxa de $P_{derrubar}$=0.1.

Suavização de etiquetasDurante o treinamento, empregamos suavização de rótulo de valor ϵ_{ls}=0.1 [36]. Isso prejudica a perplexidade, pois o modelo aprende a ser mais inseguro, mas melhora a precisão e a pontuação BLEU.

6 Resultados

6.1 Tradução Automática

Na tarefa de tradução de inglês para alemão do WMT 2014, o modelo de transformador grande (Transformer (grande) na Tabela 2) supera os melhores modelos relatados anteriormente (incluindo conjuntos) em mais de2.0 BLEU, estabelecendo uma nova pontuação BLEU de última geração de28.4.A configuração deste modelo está listada na última linha da Tabela 3. O treinamento ocorreu3.5dias em8GPUs P100. Mesmo nosso modelo básico supera todos os modelos e conjuntos publicados anteriormente, por uma fração do custo de treinamento de qualquer um dos modelos competitivos.

Na tarefa de tradução de inglês para francês do WMT 2014, nosso grande modelo atinge uma pontuação BLEU de41.0, superando todos os modelos individuais publicados anteriormente, em menos de1/4o custo de treinamento do modelo de última geração anterior. O modelo Transformer (grande) treinado para inglês para francês usou a taxa de abandono $P_{derrubar}$=0.1,em vez de0.3.

Para os modelos base, usamos um único modelo obtido pela média dos últimos 5 pontos de verificação, que foram escritos em intervalos de 10 minutos. Para os modelos grandes, calculamos a média dos últimos 20 pontos de verificação. Usamos busca de feixe com um tamanho de feixe de4e penalidade de comprimento α=0.6 [38]. Esses hiperparâmetros foram escolhidos após experimentação no conjunto de desenvolvimento. Definimos o comprimento máximo de saída durante a inferência para comprimento de entrada +50,mas terminar cedo quando possível [38].

A Tabela 2 resume nossos resultados e compara nossa qualidade de tradução e custos de treinamento com outras arquiteturas de modelo da literatura. Estimamos o número de operações de ponto flutuante usadas para treinar um modelo multiplicando o tempo de treinamento, o número de GPUs usadas e uma estimativa da capacidade de ponto flutuante de precisão única sustentada de cada GPU[5].

6.2 Variações do modelo

Para avaliar a importância de diferentes componentes do Transformer, variamos nosso modelo básico de maneiras diferentes, medindo a mudança de desempenho na tradução de inglês para alemão no

[5]Usamos valores de 2,8, 3,7, 6,0 e 9,5 TFLOPS para K80, K40, M40 e P100, respectivamente.

Tabela 3: Variações na arquitetura Transformer. Os valores não listados são idênticos aos do modelo base. Todas as métricas estão no conjunto de desenvolvimento de tradução de inglês para alemão, newstest2013. As perplexidades listadas são por palavra, de acordo com nossa codificação de par de bytes, e não devem ser comparadas às perplexidades por palavra.

	N	d_{modelo}	d_{ff}	h	d_k	d_v	P_{derrubar}	ϵ_{ls}	trem passos	PPL (dev)	AZUL (dev)	parâmetros $\times 10^6$
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)	incorporação posicional em vez de sinusóides									4,92	25,7	
grande	6	1024	4096	16			0,3		300 mil	4,33	26,4	213

conjunto de desenvolvimento, newstest2013. Usamos a busca de feixe conforme descrito na seção anterior, mas sem média de ponto de verificação. Apresentamos esses resultados na Tabela 3.

Nas linhas da Tabela 3 (A), variamos o número de cabeças de atenção e as dimensões de chave e valor de atenção, mantendo a quantidade de computação constante, conforme descrito na Seção 3.2.2. Embora a atenção de cabeça única seja 0,9 BLEU pior do que a melhor configuração, a qualidade também cai com muitas cabeças.

Nas linhas da Tabela 3 (B), observamos que reduzir o tamanho da chave de atenção$d_k$prejudica a qualidade do modelo. Isso sugere que determinar a compatibilidade não é fácil e que uma função de compatibilidade mais sofisticada do que o produto escalar pode ser benéfica. Observamos ainda nas linhas (C) e (D) que, como esperado, modelos maiores são melhores e o dropout é muito útil para evitar o ajuste excessivo. Na linha (E), substituímos nossa codificação posicional senoidal por incorporações posicionais aprendidas [9] e observamos resultados quase idênticos ao modelo básico.

6.3 Análise do grupo constituinte inglês

Para avaliar se o Transformer pode generalizar para outras tarefas, realizamos experimentos na análise de constituintes ingleses. Esta tarefa apresenta desafios específicos: a saída está sujeita a fortes restrições estruturais e é significativamente mais longa do que a entrada. Além disso, os modelos RNN sequência a sequência não foram capazes de obter resultados de ponta em regimes de dados pequenos [37].

Treinamos um transformador de 4 camadas comd_{modelo}=1024na parte do Wall Street Journal (WSJ) do Penn Treebank [25], cerca de 40 mil sentenças de treinamento. Também o treinamos em um ambiente semi-supervisionado, usando os maiores corpora de alta confiança e BerkleyParser com aproximadamente 17 milhões de sentenças [37]. Usamos um vocabulário de tokens de 16K para a configuração somente WSJ e um vocabulário de tokens de 32K para a configuração semi-supervisionada.

Realizamos apenas um pequeno número de experimentos para selecionar o dropout, atenção e residual (seção 5.4), taxas de aprendizado e tamanho do feixe no conjunto de desenvolvimento da Seção 22, todos os outros parâmetros permaneceram inalterados em relação ao modelo de tradução base de inglês para alemão. Durante a inferência, nós

Tabela 4: O Transformer generaliza bem para a análise do eleitorado inglês (os resultados estão na Seção 23 do WSJ)

analisador	Treinamento	WSJ 23 F1
Vinyals & Kaiser et al. (2014) [37]	Apenas WSJ, apenas WSJ	88,3
Petrov et ai. (2006) [29]	discriminativo, apenas WSJ	90,4
Zhu et al. (2013) [40]	discriminativo, apenas WSJ	90,4
Dyer et al. (2016) [8]	discriminativo, apenas discriminativo	91,7
Transformador (4 camadas)	WSJ apenas, discriminativo	91,3
Zhu et al. (2013) [40]	semi-supervisionado	91,3
Huang & Harper (2009) [14]	semi-supervisionado	91,3
McClosky et al. (2006) [26]	semi-supervisionado	92.1
Vinyals & Kaiser et al. (2014) [37]	semi-supervisionado	92.1
Transformador (4 camadas)	semi-supervisionado	92,7
Luong e outros. (2015) [23]	multitarefa	93,0
Dyer et ai. (2016) [8]	generativo	93,3

aumentou o comprimento máximo de saída para o comprimento de entrada +300.Usamos um tamanho de feixe de21eα=0.3 apenas para WSJ e para o ambiente semi-supervisionado.

Nossos resultados na Tabela 4 mostram que, apesar da falta de ajuste específico da tarefa, nosso modelo funciona surpreendentemente bem, produzindo melhores resultados do que todos os modelos relatados anteriormente, com exceção da Gramática da Rede Neural Recorrente [8].

Em contraste com os modelos RNN sequência a sequência [37], o Transformer supera o Berkeley-Parser [29] mesmo quando treinado apenas no conjunto de treinamento WSJ de 40K sentenças.

7. Conclusão

Neste trabalho, apresentamos o Transformer, o primeiro modelo de transdução de sequência baseado inteiramente na atenção, substituindo as camadas recorrentes mais comumente utilizadas em arquiteturas codificador-decodificador por autoatenção multicabeças.

Para tarefas de tradução, o Transformer pode ser treinado significativamente mais rápido do que as arquiteturas baseadas em camadas recorrentes ou convolucionais. Nas tarefas de tradução de inglês para alemão do WMT 2014 e de inglês para francês do WMT 2014, alcançamos um novo estado da arte. Na tarefa anterior, nosso melhor modelo supera até mesmo todos os conjuntos relatados anteriormente.

Estamos entusiasmados com o futuro dos modelos baseados em atenção e planejamos aplicá-los a outras tarefas. Planejamos estender o Transformer a problemas envolvendo modalidades de entrada e saída diferentes de texto e investigar mecanismos locais de atenção restrita para lidar eficientemente com grandes entradas e saídas, como imagens, áudio e vídeo. Tornar a geração menos sequencial é outro dos nossos objetivos de pesquisa.

O código que usamos para treinar e avaliar nossos modelos está disponível emhttps://github.com/tensorflow/tensor2tensor.

ReconhecimentosSomos gratos a Nal Kalchbrenner e Stephan Gouws por seus frutíferos comentários, correções e inspiração.

Referências

[1] Jimmy Lei Ba, Jamie Ryan Kiros e Geoffrey E Hinton. Normalização de camadas.*arXiv pré-impressão arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho e Yoshua Bengio. Tradução automática neural aprendendo em conjunto a alinhar e traduzir.*CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong e Quoc V. Le. Exploração massiva de arquiteturas de tradução automática neural.*CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong e Mirella Lapata. Redes de memória de longo prazo para leitura de máquina.*arXiv pré-impressão arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk e Yoshua Bengio. Aprendendo representações de frases usando codificador-decodificador rnn para tradução automática estatística.*CoRR*, abs/1406.1078, 2014.

[6] François Chollet. Xception: Deep Learning com convoluções separáveis em profundidade.*arXiv pré-impressão arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho e Yoshua Bengio. Avaliação empírica de redes neurais recorrentes fechadas na modelagem de sequências.*CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros e Noah A. Smith. Gramáticas de redes neurais recorrentes. Em*Proc. de NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats e Yann N. Dauphin. Sequência convolucional para aprendizado de sequência.*arXiv pré-impressão arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Geração de sequências com redes neurais recorrentes.*arXiv pré-impressão arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun. Aprendizagem residual profunda para reconhecimento de imagem. Em*Anais da Conferência IEEE sobre Visão Computacional e Reconhecimento de Padrões*, páginas 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi e Jürgen Schmidhuber. Fluxo gradiente em redes recorrentes: a dificuldade de aprender dependências de longo prazo, 2001.

[13] Sepp Hochreiter e Jürgen Schmidhuber. Memória de longo prazo.*computação neural*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang e Mary Harper. Gramáticas PCFG de autotreinamento com anotações latentes em vários idiomas. Em*Anais da Conferência de 2009 sobre Métodos Empíricos em Processamento de Linguagem Natural*, páginas 832–841. ACL, agosto de 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer e Yonghui Wu. Explorando os limites da modelagem de linguagem.*arXiv pré-impressão arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser e Samy Bengio. A memória ativa pode substituir a atenção? Em*Avanços em Sistemas de Processamento de Informação Neural, (NIPS)*, 2016.

[17] Łukasz Kaiser e Ilya Sutskever. GPUs neurais aprendem algoritmos. Em*Conferência Internacional sobre Representações de Aprendizagem (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves e Koray Kavukcuoglu. Tradução automática neural em tempo linear.*arXiv pré-impressão arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang e Alexander M. Rush. Redes de atenção estruturadas. Em*Conferência Internacional sobre Representações de Aprendizagem*, 2017.

[20] Diederik Kingma e Jimmy Ba. Adam: Um método para otimização estocástica. Em*ICLR*, 2015.

[21] Oleksii Kuchaiev e Boris Ginsburg. Truques de fatoração para redes LSTM.*arXiv pré-impressão arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou e Yoshua Bengio. Uma incorporação de frase autoatentiva estruturada.*arXiv pré-impressão arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals e Lukasz Kaiser. Sequência multitarefa para aprendizado de sequência.*arXiv pré-impressão arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham e Christopher D Manning. Abordagens eficazes para tradução automática neural baseada em atenção.*arXiv pré-impressão arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz e Beatrice Santorini. Construindo um grande corpus anotado de inglês: The penn treebank.*linguística computacional*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak e Mark Johnson. Autotreinamento eficaz para análise. Em *Anais da Conferência de Tecnologia da Linguagem Humana da NAACL, Conferência Principal*, páginas 152–159. ACL, junho de 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das e Jakob Uszkoreit. Um modelo de atenção decomponível. Em *Métodos Empíricos em Processamento de Linguagem Natural*, 2016.

[28] Romain Paulus, Caiming Xiong e Richard Socher. Um modelo profundamente reforçado para sumarização abstrativa.*arXiv pré-impressão arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux e Dan Klein. Aprendendo anotação de árvore precisa, compacta e interpretável. Em*Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, páginas 433–440. ACL, julho de 2006.

[30] Ofir Press e Lior Wolf. Usando a incorporação de saída para melhorar os modelos de linguagem.*arXiv pré-impressão arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow e Alexandra Birch. Tradução automática neural de palavras raras com unidades de subpalavras.*arXiv pré-impressão arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton e Jeff Dean. Redes neurais escandalosamente grandes: a camada de mistura de especialistas com portas esparsas.*arXiv pré-impressão arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov. Dropout: uma maneira simples de evitar o overfitting das redes neurais.*Jornal de pesquisa de aprendizado de máquina*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston e Rob Fergus. Redes de memória fim-a-fim. Em C. Cortes, ND Lawrence, DD Lee, M. Sugiyama e R. Garnett, editores, *Avanços em Sistemas de Processamento de Informação Neural 28*, páginas 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals e Quoc VV Le. Aprendizado de sequência a sequência com redes neurais. Em *Avanços em Sistemas de Processamento de Informação Neural*, páginas 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens e Zbigniew Wojna. Repensando a arquitetura inicial para visão computacional.*CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever e Hinton. A gramática como língua estrangeira. Em *Avanços em Sistemas de Processamento de Informação Neural*, 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 neural de tradução automática do Google: fazendo a ponte entre a tradução humana e a automática.*arXiv pré-impressão arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li e Wei Xu. Modelos recorrentes profundos com conexões rápidas para tradução automática neural.*CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang e Jingbo Zhu. Análise de constituintes de redução de deslocamento rápida e precisa. Em*Actas da 51ª Reunião Anual do ACL (Volume 1: Long Papers)*, páginas 434–443. ACL, agosto de 2013.

Visualizações de atenção

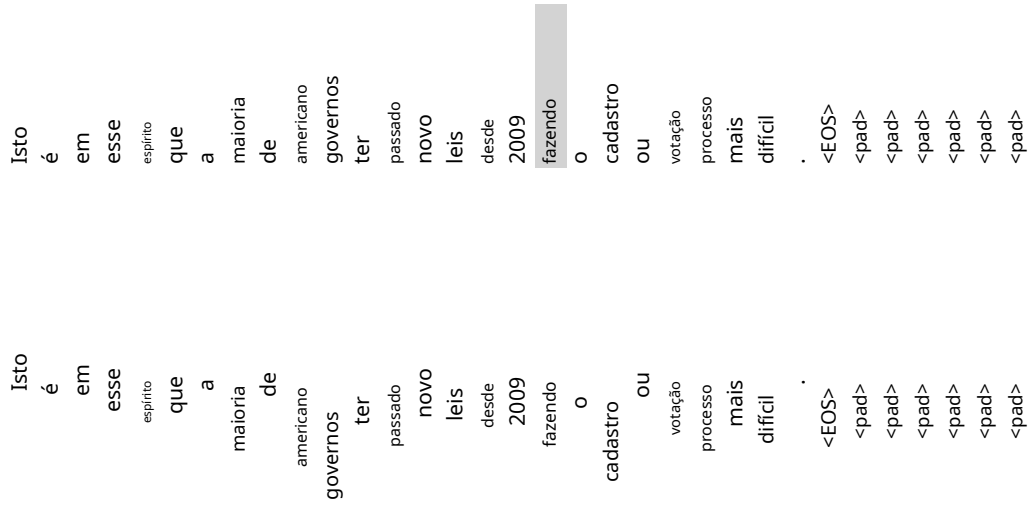
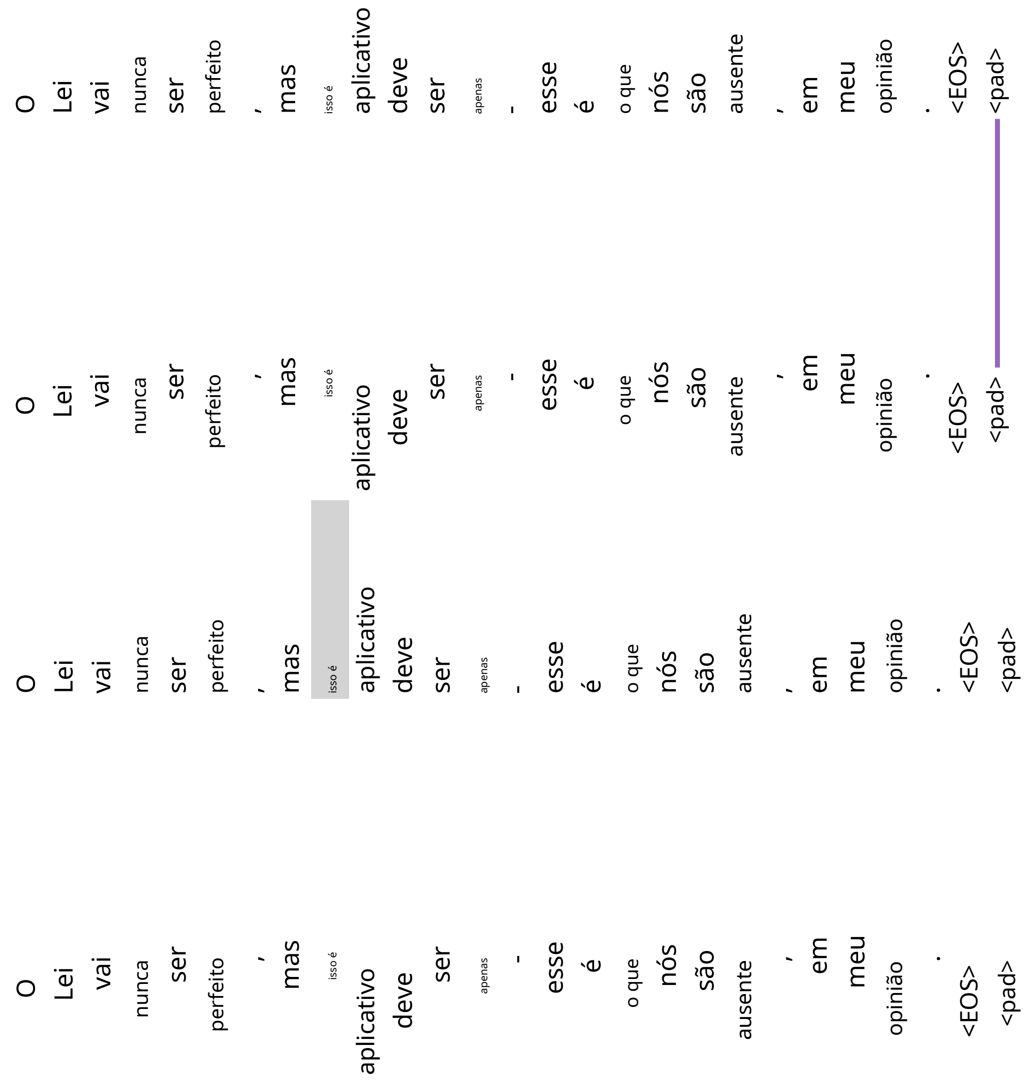


Figura 3: Um exemplo do mecanismo de atenção seguindo dependências de longa distância na auto-atenção do codificador na camada 5 de 6. Muitas das cabeças de atenção atendem a uma dependência distante do verbo 'fazer', completando a frase 'fazer.. .mais difícil'. Atenções aqui mostradas apenas para a palavra 'fazer'. Cores diferentes representam cabeças diferentes. Melhor visualizado em cores.



Fegurė 4: Twaett ntisobre ele de Anúnciostambém s, a enhayer 5 o f 6 a, parentemente envolvido na resolução da anáfora. Principal: Fcompleto ntopttionsparaele de Anúncios 5. bottom: É olatedndeizio ns de apenas a palavra 'its' para atenção encabeça 5 s and 6. Note isso the aett ntisobre s rayeh er s arpf ou isso palavra.

Figure 5: Muitas das cabeças de atenção exibem um comportamento que parece relacionado à estrutura do sentptce. Ceig eudoisuch exam por esalo eu, fromtduas cabeças diferentes do codificador auto-atenção atl avórs 5 de 6. º e heads clearlvéseam el paperfem diferentes tarefas.