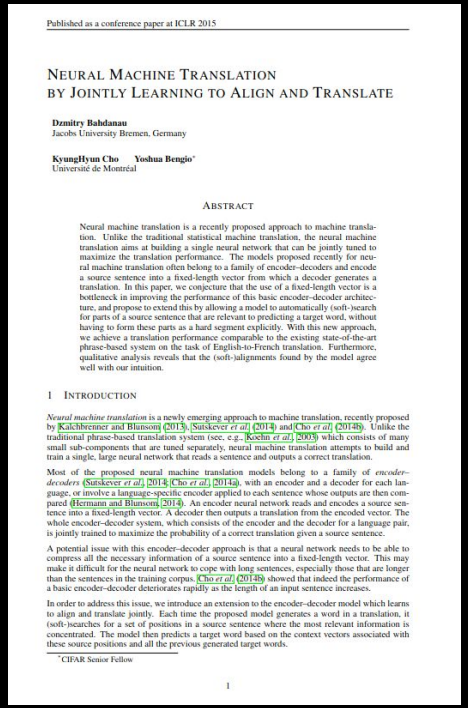
Transformers

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR



Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.

In order to address this issue, we introduce an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

*CIFAR Senior Fellow

Bahdanau, Cho e Bengio. **Neural machine translation by jointly learning to align and translate**. arXiv preprint arXiv:1409.0473, 2014.

Attention Is All You Need

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* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

*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.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser e Polosukhin. **Attention is all you need.** Advances in Neural Information Processing Systems, 2017.

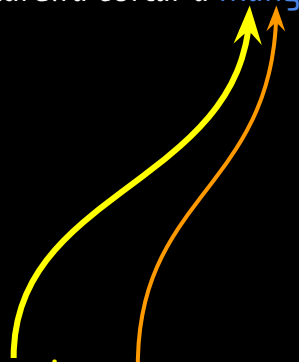
Mecanismo de Atenção

Eu gosto de comer manga.

Ele pediu para a costureira cortar a manga.

Mecanismo de Atenção

Ele pediu para a costureira cortar a manga.



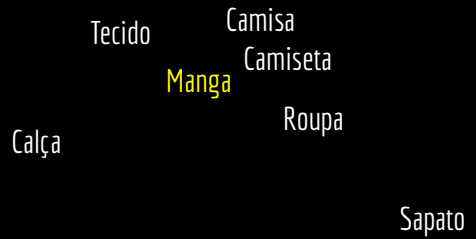
Ele pediu para a costureira cortar a manga.

A rede precisa prestar atenção em determinadas palavras, dependendo da sentença.

Transformer

Eu gosto de comer manga.

Ele pediu para a costureira cortar a manga.



Um transformer mapeia um vetor para uma região do espaço, dependendo dos demais vetores ao seu redor.

Problema em um MLP Convencional

Eu gosto de comer manga.

Quarto componente da sentença é um dos mais importantes.

Ele pediu para a costureira cortar a manga.

Quinto e sexto componentes da sentença são mais importantes.

Problema em um MLP Convencional

Eu gosto de comer manga.

Quarto componente da sentença é um dos mais importantes.

Ele pediu para a costureira cortar a manga.

Quinto e sexto componentes da sentença são mais importantes.

Se as frases são entradas de uma rede (características), o peso do quarto neurônio deveria ser maior quando entrando com a primeira frase (e do quinto e sexto da segunda frase).

Mas os pesos são fixos depois do treinamento em um MLP convencional.

Entrada

Entrada de um transformer.

Conjunto de vetores $\{\mathbf{x}_n\}$ de D dimensões, com $n = 1, ..., N$.

Vetores chamados de *tokens*.

Cada *token* pode ser uma palavra em um frase, um pedaço de uma imagem, um trecho de áudio, ...

Entrada

Entrada de um transformer.

Conjunto de vetores $\{\boldsymbol{x}_n\}$ de D dimensões, com $n = 1, ..., N$.

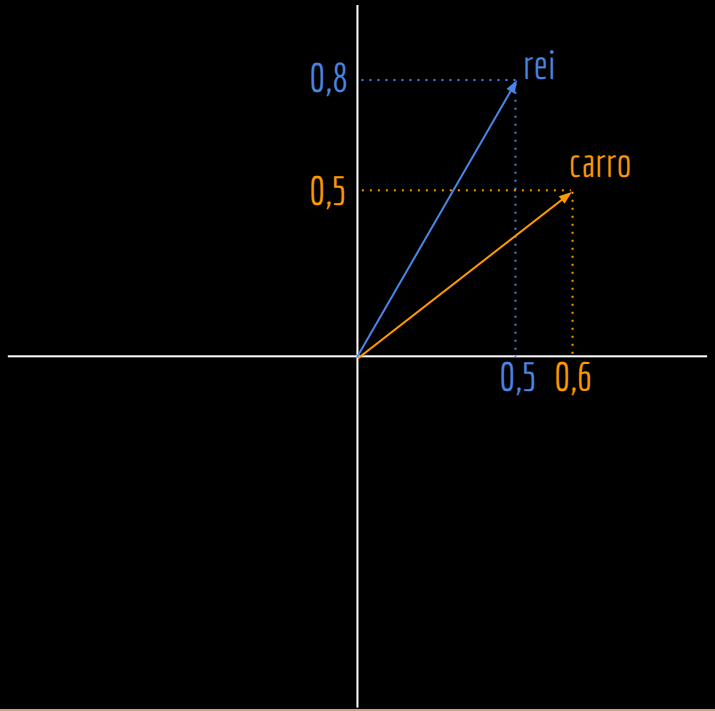
Vetores chamados de *tokens*.

Cada *token* pode ser uma palavra em um frase, um pedaço de uma imagem, um trecho de áudio, ...

Cada elemento $\boldsymbol{x}_{ni}$ do é uma característica do *token*.

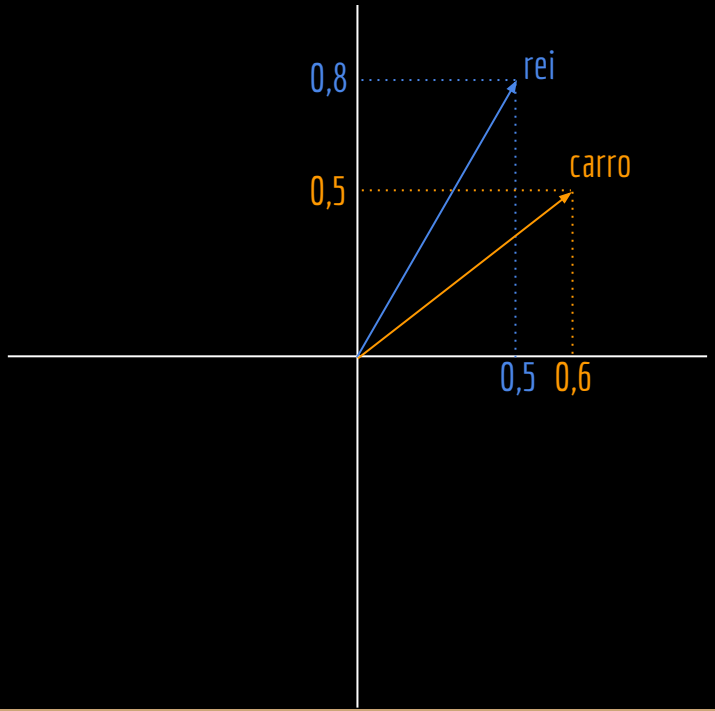
Exemplo: se $\boldsymbol{x}_n$ é um pedaço de uma imagem, cada $\boldsymbol{x}_{ni}$ pode ser um pixel.

Vetor x_n - Exemplo com Palavras



O espaço onde os vetores residem é chamado de espaço de *embeddings*.

Vetor x_n - Exemplo com Palavras



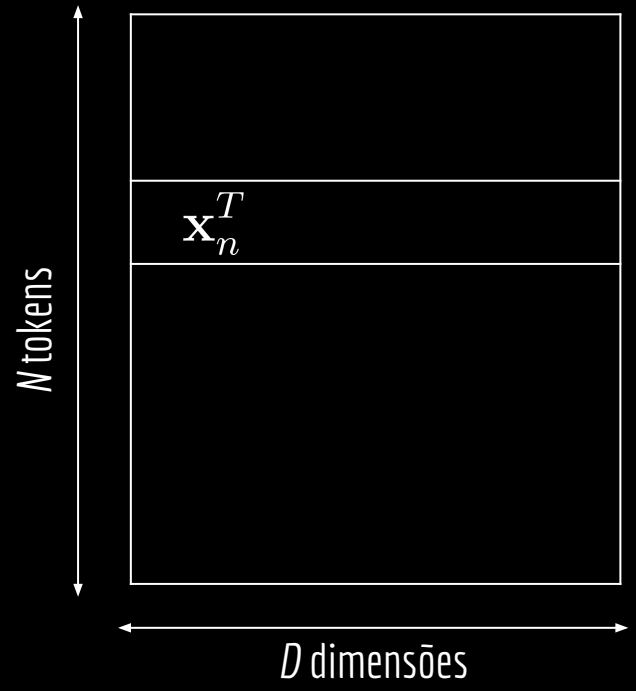
O espaço onde os vetores residem é chamado de espaço de *embeddings*.

No Llama 3 esse espaço tem 4.096, 8.192 ou 16.384 dimensões, dependendo da versão do modelo.

No GPT-3 são 12.288 dimensões.

Entrada

A entrada é uma matriz ***X*** de *N x D*, onde cada linha possui um token.



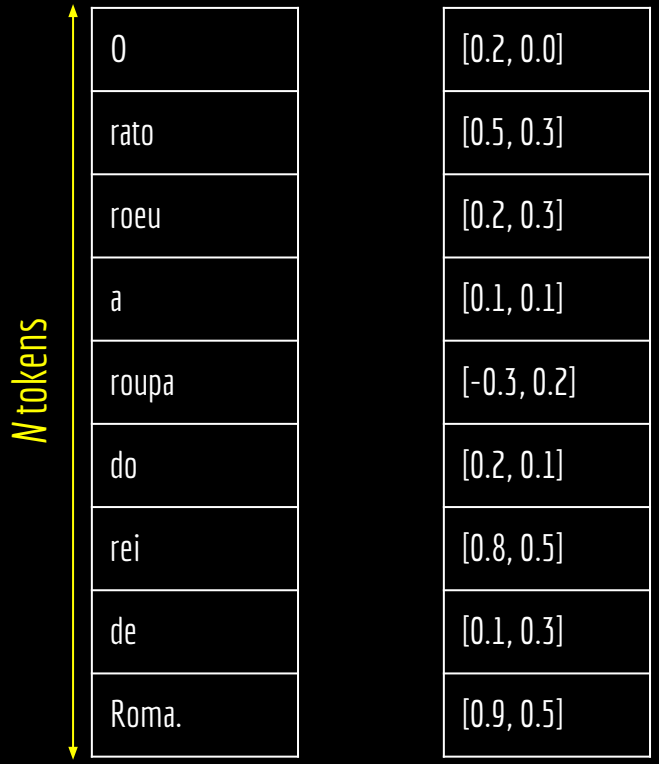
n-ésimo token de ***X*** contendo *D* dimensões.

Entrada

O	[0.2, 0.0]
rato	[0.5, 0.3]
roeu	[0.2, 0.3]
a	[0.1, 0.1]
roupa	[-0.3, 0.2]
do	[0.2, 0.1]
rei	[0.8, 0.5]
de	[0.1, 0.3]
Roma.	[0.9, 0.5]

Entrada

No GPT-3 são 2.048 tokens.
No Llama 3, são 8.192 tokens.
No Llama 3.1 em diante, 128K tokens.



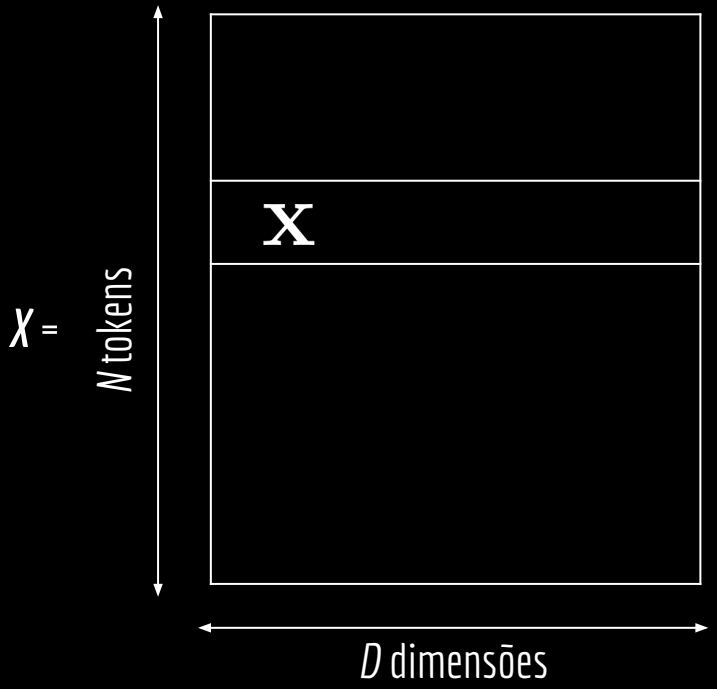
Transformer

$$\tilde{\mathbf{X}} = transformer[\mathbf{X}]$$

Um transformer pode ser considerado uma camada de uma rede.

Toma uma entrada $\mathbf{X}$ e transforma em $\tilde{\mathbf{X}}$, com a mesma dimensão de $\mathbf{X}$.

Considere



O que a operação a seguir representa?

$$Y = \mathbf{X}\mathbf{X}^T$$

Qual a dimensão da resposta da operação?

Considere

$$Y = \mathbf{X}\mathbf{X}^T$$

$$Y = \begin{vmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1d} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ X_{n1} & X_{n2} & X_{n3} & \dots & X_{nd} \end{vmatrix} \begin{vmatrix} X_{11} & X_{21} & X_{31} & \dots & X_{n1} \\ X_{12} & X_{22} & X_{32} & \dots & X_{n2} \\ X_{13} & X_{23} & X_{33} & & X_{n3} \\ \dots & \dots & \dots & \dots & \dots \\ X_{1d} & X_{2d} & X_{3d} & \dots & X_{nd} \end{vmatrix}$$

Considere

$$Y = \mathbf{X}\mathbf{X}^T$$

$$Y = \begin{vmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1d} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ X_{n1} & X_{n2} & X_{n3} & \dots & X_{nd} \end{vmatrix} \begin{vmatrix} X_{11} & X_{21} & X_{31} & \dots & X_{n1} \\ X_{12} & X_{22} & X_{32} & \dots & X_{n2} \\ X_{13} & X_{23} & X_{33} & & X_{n3} \\ \dots & \dots & \dots & \dots & \dots \\ X_{1d} & X_{2d} & X_{3d} & \dots & X_{nd} \end{vmatrix} = \begin{vmatrix} X_1 X_1^T & & & \\ & & & \\ & & & \\ & & & \end{vmatrix}$$

Considere

$$Y = \mathbf{X}\mathbf{X}^T$$

$$Y = \begin{vmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1d} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ X_{n1} & X_{n2} & X_{n3} & \dots & X_{nd} \end{vmatrix} \begin{vmatrix} X_{11} & X_{21} & X_{31} & \dots & X_{n1} \\ X_{12} & X_{22} & X_{32} & \dots & X_{n2} \\ X_{13} & X_{23} & X_{33} & & X_{n3} \\ \dots & \dots & \dots & \dots & \dots \\ X_{1d} & X_{2d} & X_{3d} & \dots & X_{nd} \end{vmatrix} = \begin{vmatrix} X_1X_1^T & X_1X_2^T \\ & \\ & \\ & \end{vmatrix}$$

Considere

$$Y = \mathbf{X}\mathbf{X}^T$$

$$Y = \begin{vmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1d} \\ X_{21} & X_{22} & X_{23} & \dots & X_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ X_{n1} & X_{n2} & X_{n3} & \dots & X_{nd} \end{vmatrix} \begin{vmatrix} X_{11} & X_{21} & X_{31} & \dots & X_{n1} \\ X_{12} & X_{22} & X_{32} & \dots & X_{n2} \\ X_{13} & X_{23} & X_{33} & & X_{n3} \\ \dots & \dots & \dots & \dots & \dots \\ X_{1d} & X_{2d} & X_{3d} & \dots & X_{nd} \end{vmatrix} = \begin{vmatrix} X_1X_1^T & X_1X_2^T & X_1X_3^T \\ & & \\ & & \\ & & \end{vmatrix}$$

Considere

$$Y = \mathbf{X}\mathbf{X}^T$$

$$Y = \begin{vmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1d} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nd} \end{vmatrix} \begin{vmatrix} x_{11} & x_{21} & x_{31} & \dots & x_{n1} \\ x_{12} & x_{22} & x_{32} & \dots & x_{n2} \\ x_{13} & x_{23} & x_{33} & & x_{n3} \\ \dots & \dots & \dots & \dots & \dots \\ x_{1d} & x_{2d} & x_{3d} & \dots & x_{nd} \end{vmatrix} = \begin{vmatrix} x_1x_1^T & x_1x_2^T & x_1x_3^T & \dots & x_1x_n^T \\ x_2x_1^T & x_2x_2^T & x_2x_3^T & \dots & x_2x_n^T \\ x_3x_3^T & x_3x_2^T & x_3x_3^T & \dots & x_3x_n^T \\ \dots & \dots & \dots & \dots & \dots \\ x_nx_1^T & x_nx_2^T & x_nx_3^T & \dots & x_nx_n^T \end{vmatrix}$$

$N \times D$ $\quad$ $D \times N$ $\quad$ $N \times N$

Considere

$$Y = \mathbf{X}\mathbf{X}^T$$

Produto escalar entre cada um dos *tokens*.

Valores maiores indicam *tokens* que apontam na mesma direção.

"Atendem" mais a determinado vetor.

Podemos usar esses coeficientes como **coeficientes de atenção**.

Softmax

Desejamos que:

- A soma dos coeficientes seja 1.
- Um coeficiente grande positivo não possa ser compensado por um pequeno negativo.
- Não podemos prestar atenção em tudo ao mesmo tempo.

    Se prestarmos mais atenção em um *token*, devemos prestar menos atenção em outro.

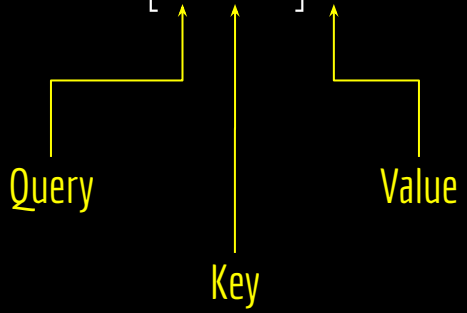
Normalizamos via *softmax[.]*, onde cada linha da matriz será normalizada para somar 1.

$$Y = softmax[\mathbf{X}\mathbf{X}^T]$$

Self-Attention

Podemos usar os coeficientes de atenção para calcular suas influências em **X**.

$$Y = softmax[\mathbf{X}\mathbf{X}^T]\mathbf{X}$$



Processo chamado de auto atenção (*self-attention*).

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Podemos definir as seguintes matrizes:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Podemos definir as seguintes matrizes:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

Cada $\mathbf{W}^{(.)}$ é uma matriz de pesos que deve ser ajustada durante o treino.

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Podemos definir as seguintes matrizes:

$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$ Qual os pesos dos vetores de entrada na forma de queries.

$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$

$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Podemos definir as seguintes matrizes:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$$

Como esses vetores devem influenciar as chaves (qual a atenção dada a cada chave?).

$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Podemos definir as seguintes matrizes:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

Como a matriz de valores possíveis de saída é afetada pela combinação de **Q** e **K**?

Problema

A operação $\mathbf{X}\mathbf{X}^T$ é fixa.

Podemos definir as seguintes matrizes:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$$

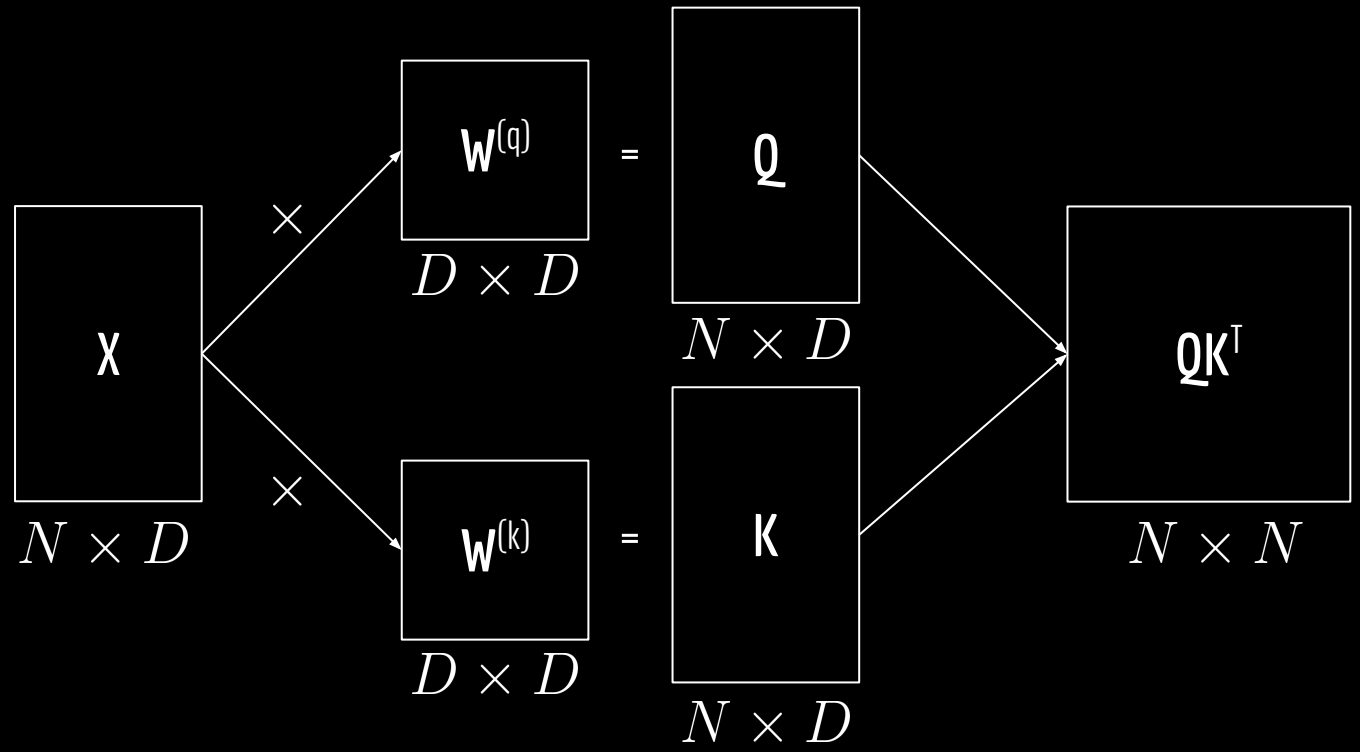
$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

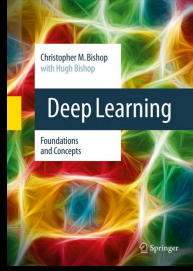
$$Y = softmax[\mathbf{Q}\mathbf{K}^T]\mathbf{V}$$

Processo

$$Y = softmax[\mathbf{Q}\mathbf{K}^T]\mathbf{V}$$



Bishop, C. M., Bishop, H. Deep Learning: Foundations and Concepts. 2023.



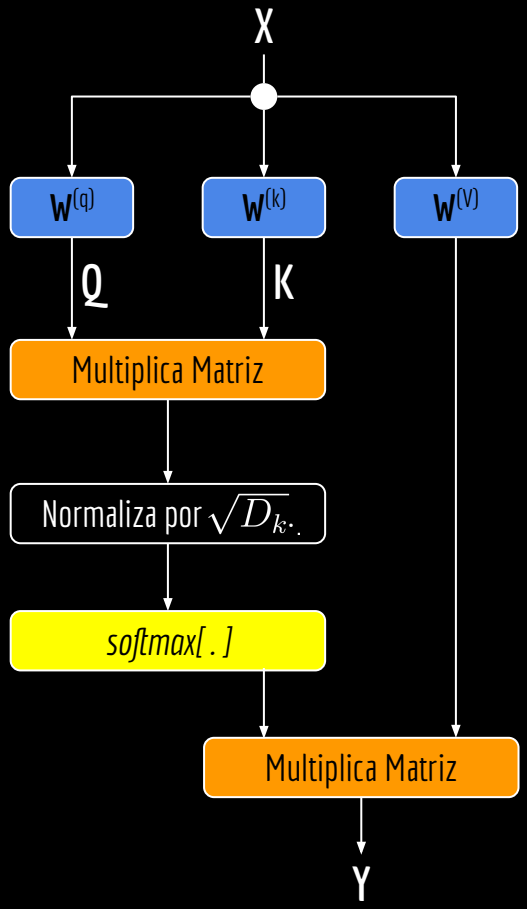
Normalizando

Finalmente, podemos normalizar os dados de entrada do softmax usando o tamanho D_k da matriz $\mathbf{W}^{(k)}$ (que possui tamanho $D \times D_k$).

Assumindo que os vetores das matrizes de query e key são independentes, com média zero e variância um, podemos:

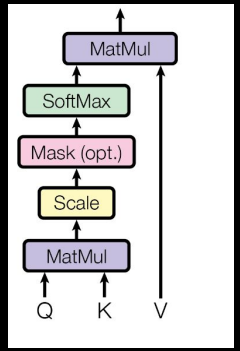
$$Y = softmax\left[\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}\right]\mathbf{V}$$

Self-attention head

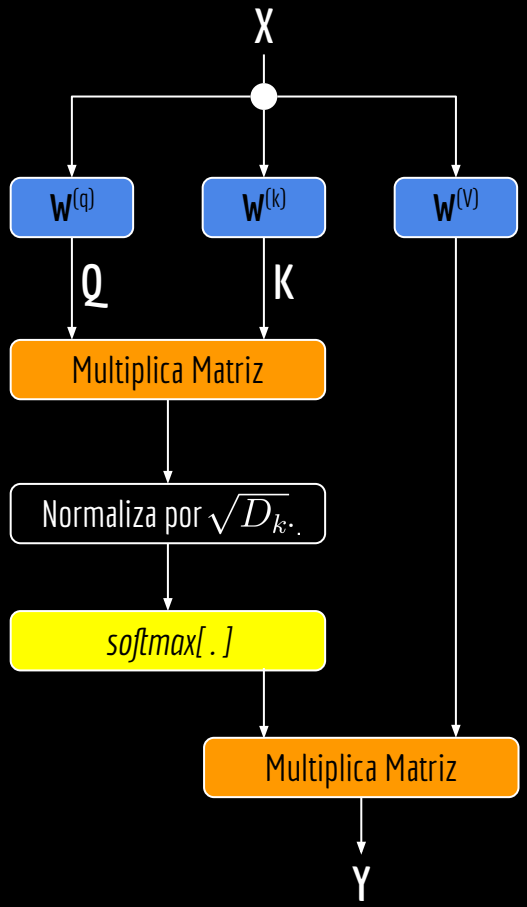


Self-attention head

Ideia básica das cabeças de atenção dos modelos LLama, GPT, Bert, ...

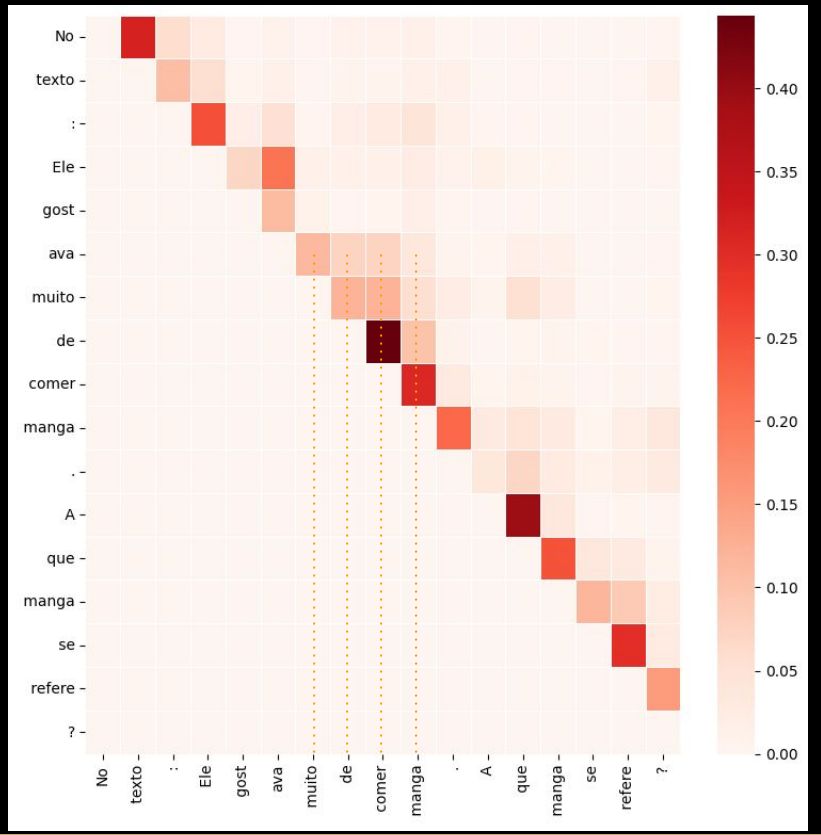


Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser e Polosukhin. **Attention is all you need**. Advances in Neural Information Processing Systems, 2017.



Heatmap de uma cabeça

“No texto: Ele gostava muito de comer manga. A que manga se refere?”

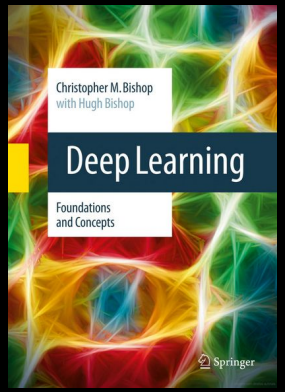


Exercícios

1. Verifique os mapas de calor gerados a partir das cabeças de atenção do modelo Llama 3.2 disponibilizado no Google Colab.
    a. Existem trechos comentados no exemplo onde é tirada a média das cabeças da última camada, e onde é utilizada uma cabeça específica da última camada.
        i. Faça testes com outras camadas, cabeças, e frases.

Referências

Bishop, C. M., Bishop, H. Deep Learning: Foundations and Concepts. 2023.



Bahdanau, Cho e Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.


Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho Yoshua Bengio*
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder-decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever et al. (2014), and Cho et al. (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn et al., 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

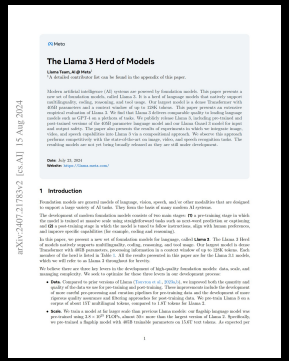
Most of the proposed neural machine translation models belong to a family of encoder-decoders (Sutskever et al., 2014; Cho et al., 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2013). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder-decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder-decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho et al. (2014b) showed that indeed the performance of a basic encoder-decoder deteriorates rapidly as the length of an input sentence increases.

In order to address this issue, we introduce an extension to the encoder-decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

*CIFAR Senior Fellow

Dubey, A. et al. The Llama 3 Herd of Models. 2024.



Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser e Polosukhin. Attention is all you need. Advances in Neural Information Processing Systems, 2017.


Attention Is All You Need

Ashish Vaswani* Noam Shazeer* Niki Parmar* Jakob Uszkoreit*
Google Brain Google Brain Google Research Google Research
avaswani@google.com noam@google.com nikip@google.com usz@google.com

Llion Jones* Aidan N. Gomez* Łukasz Kaiser*
Google Research University of Toronto Google Brain
llion@google.com aidan@cs.toronto.edu lukaszkaiser@google.com

Illia Polosukhin*
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

*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.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Licença