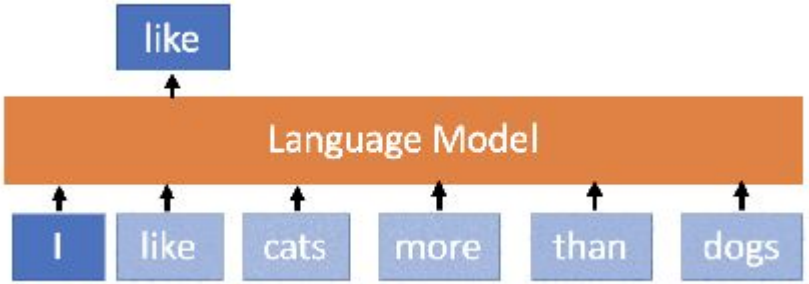
Transformers

Introdução

O que é um modelo de língua?

- Objetivo: Prever um termo dado um contexto
- Aplicações
  - Autocomplete de Texto
  - Transfer Learning
- Tipos de modelos de língua
  - Probabilísticos
  - Sequências
  - Atenção



Modelos Probabilísticos

É um modelo que visa estimar a probabilidade de um termo dado um contexto.

Exemplo de modelos: N Gramas, Word2Vec

Problemas:

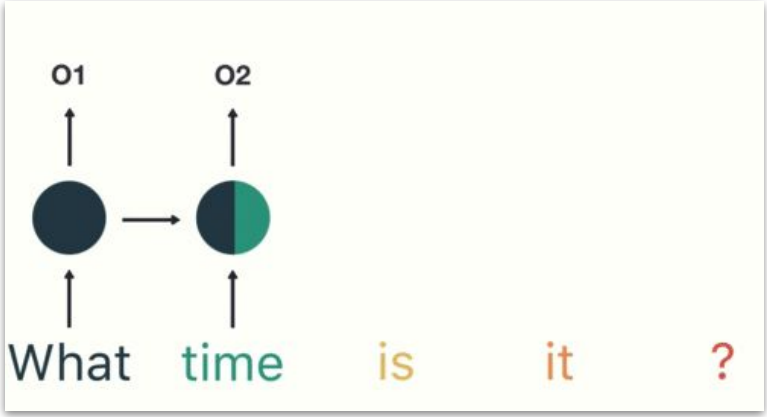
- Baixa complexidade
- Não levam em conta a sequência das palavras

P(próxima palavra) = P(Termo | O Grupo Turing é)

Termo	Probabilidade
Legal	60%
...	...
Abacaxi	0.01%

Modelos Sequenciais

Redes neurais recorrentes (RNN) são uma classe de redes neurais poderosa para modelar dados de sequência, como séries temporais ou linguagem natural.



Problemas:

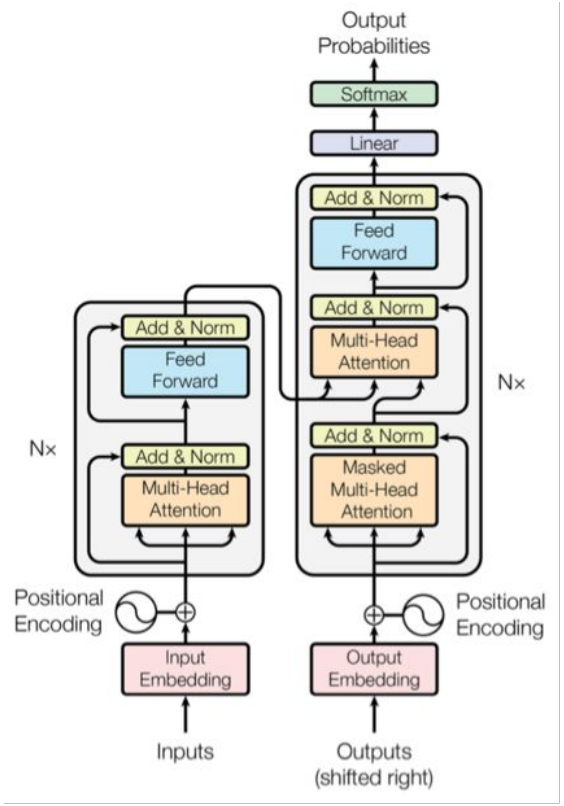
- Processamento Sequencial
- Perda de informação com textos extensos
- Vanishing Gradients

Modelos de Atenção

Classe de rede neurais poderosa para modelar sequências paralelamente, utilizando mecanismos de atenção.

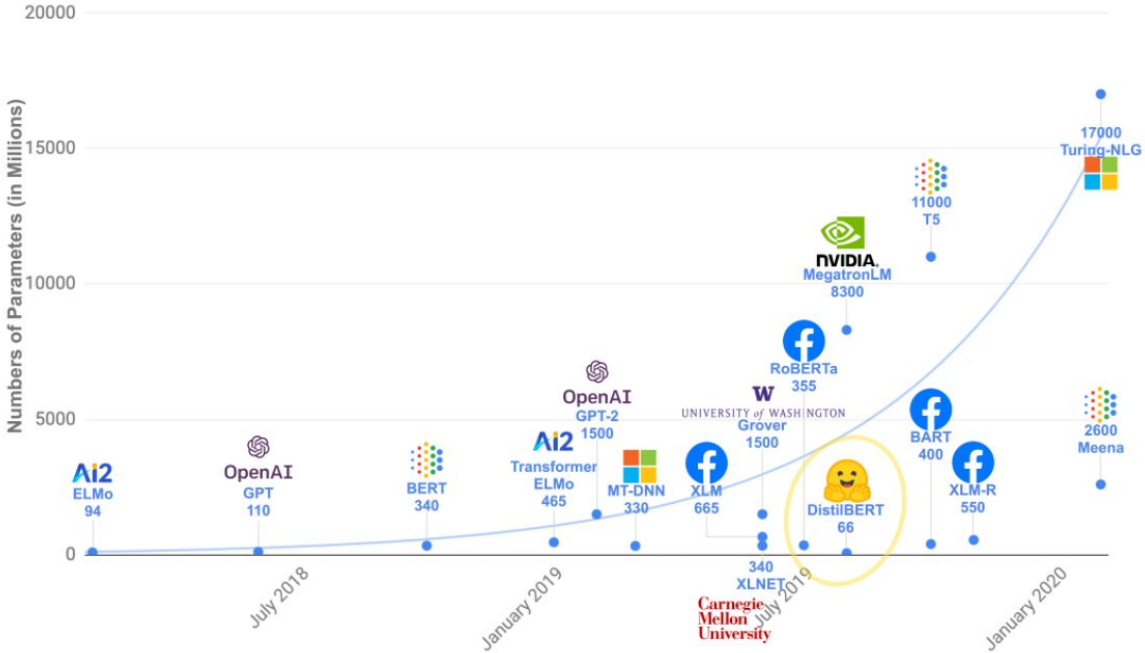
Problema:

- Custo e Tempo de treinamento



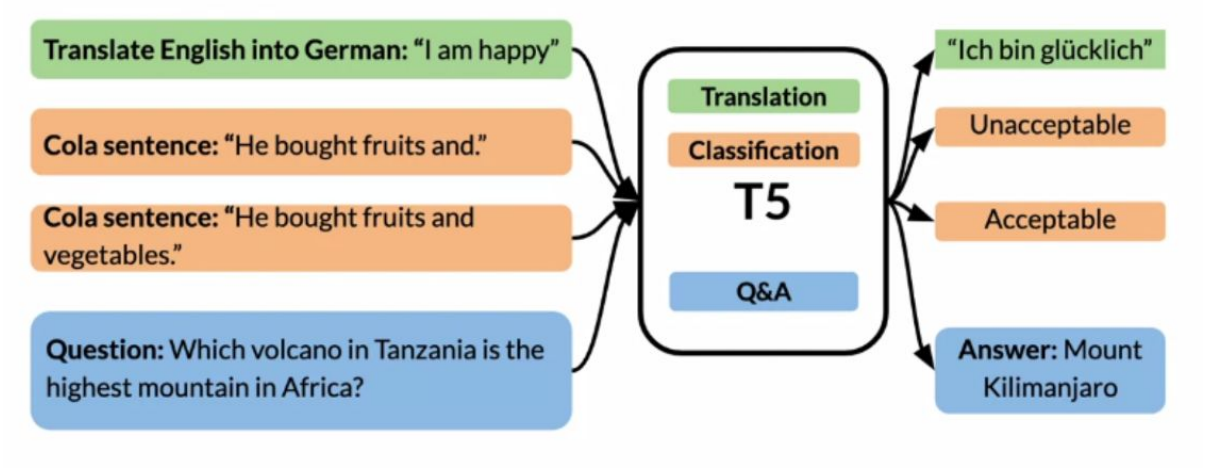
Modelos de Atenção

Muitos parâmetros



Exemplo de Transformers: T5 do Google

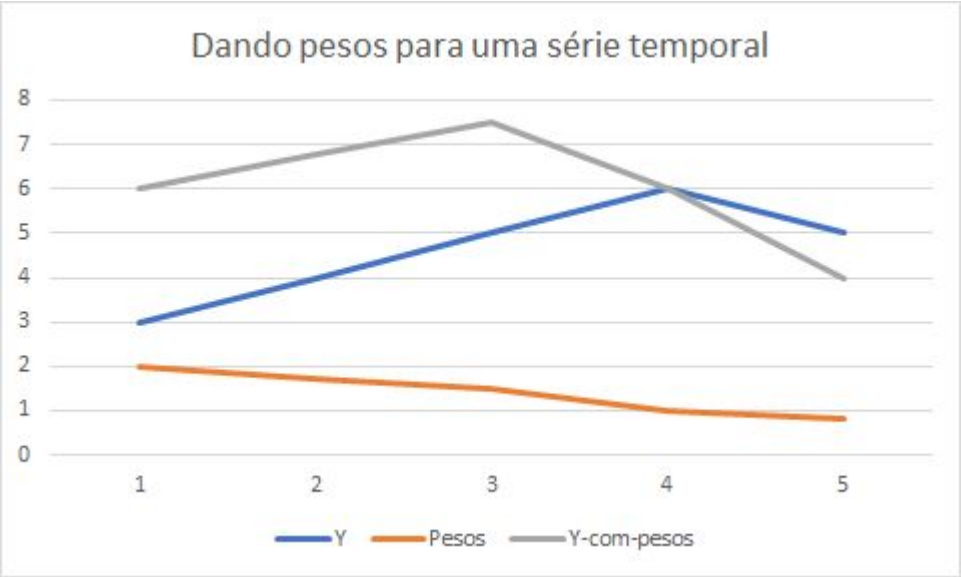
Podem ser treinados para executar mais de uma tarefa.



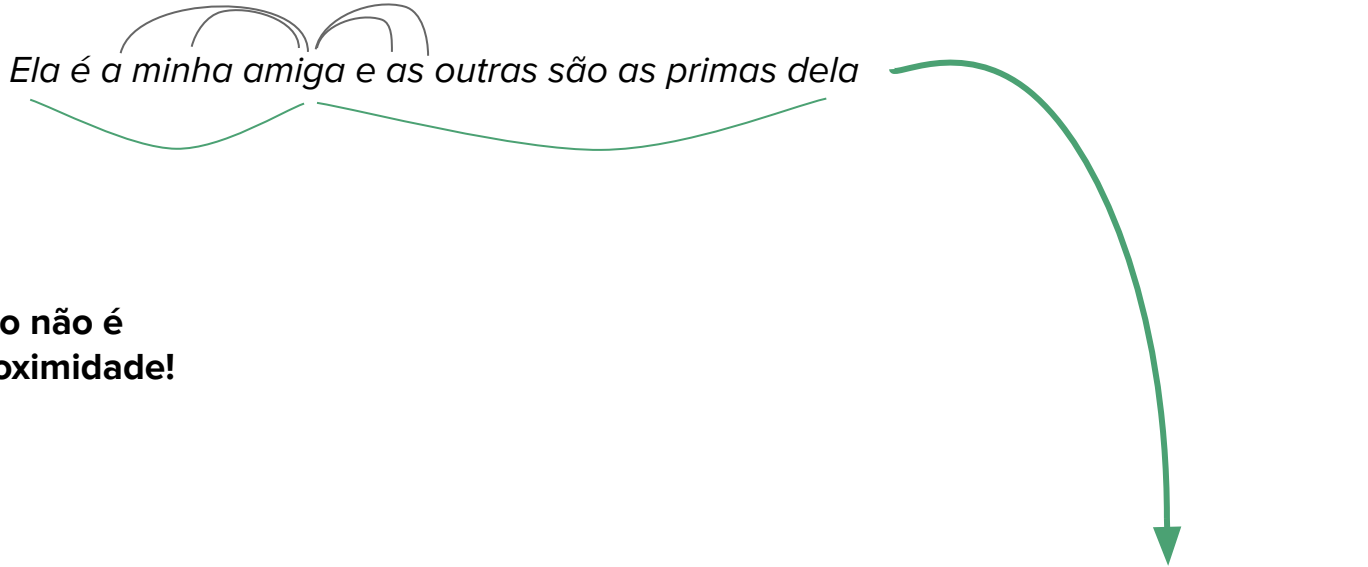
Self Attention

Mecanismos de Atenção

Y	Pesos	Y-com-pesos
3	2	6
4	1.7	6.8
5	1.5	7.5
6	1	6
5	0.8	4



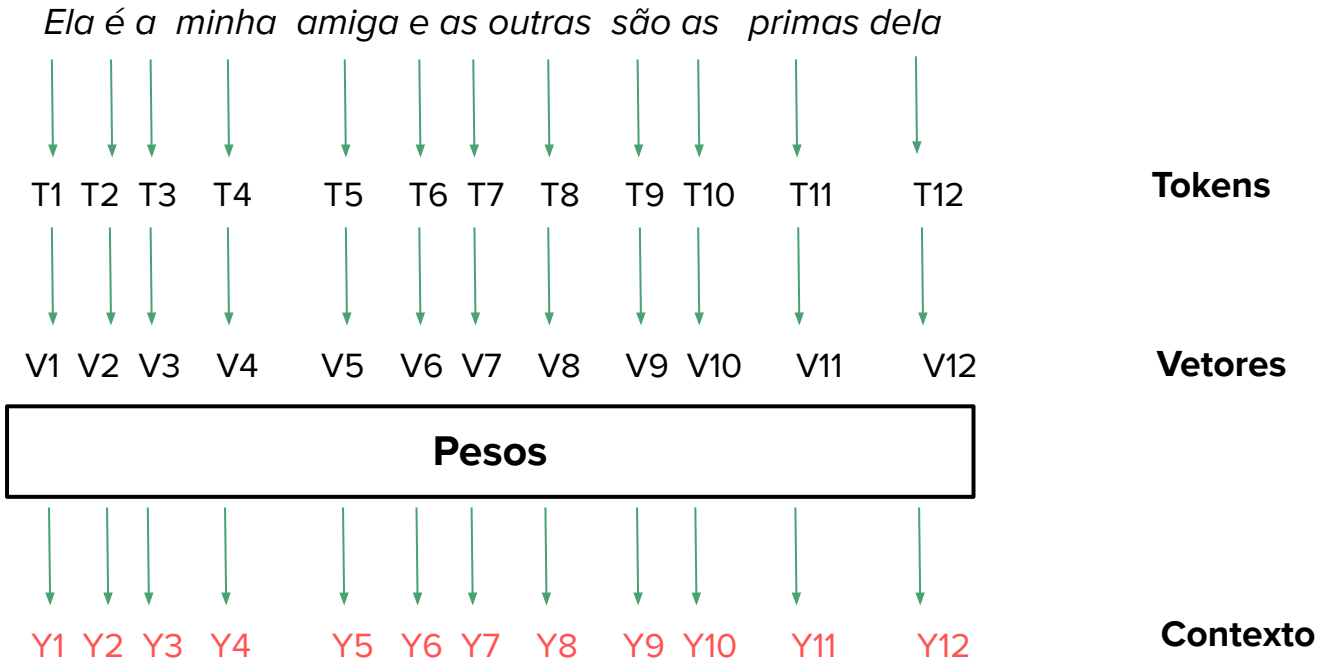
Self Attention - Introdução



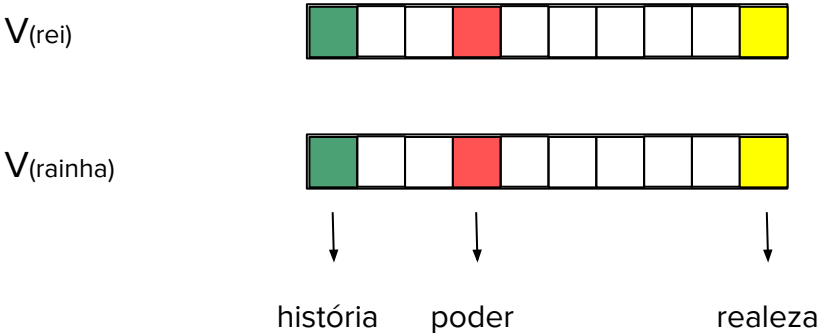
Significado linguístico não é determinado pela proximidade!

Como podemos encontrar esse relacionamento de forma automática?

Self Attention - Introdução



Recapitulando - Word embeddings



Os números não representam essas propriedades *diretamente*, mas podemos pensar nesses valores como uma codificação delas

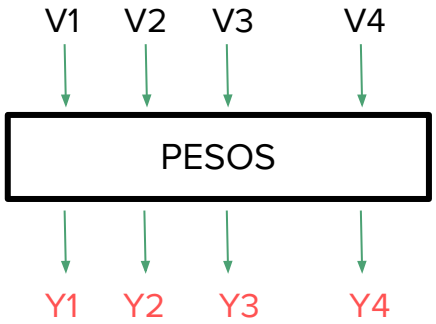
- castelo
- país
- riqueza
- coroa

- avião
- computador
- chinelo
- óculos

Compartilham significado mesmo sem proximidade!

Self Attention

Manga da minha camiseta



Versão contextualizada

$V_1 \cdot V_1 = W_{11}$

$V_1 \cdot V_2 = W_{12}$

$V_1 \cdot V_3 = W_{13}$

$V_1 \cdot V_4 = W_{14}$

normalização (soma = 1) →

W_{11}

W_{12}

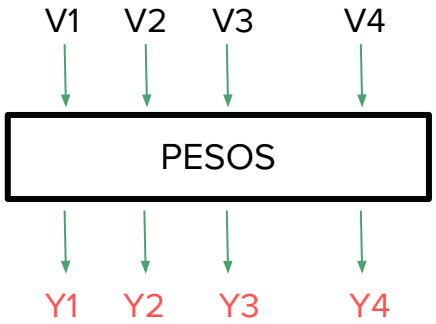
W_{13}

W_{14}

$$W_{11}V_1 + W_{12}V_2 + W_{13}V_3 + W_{14}V_4 = Y_1$$

Self Attention

Manga da minha camiseta



$V_1 \; V_1 = W_{11}$

$V_1 \; V_2 = W_{12}$

$V_1 \; V_3 = W_{13}$

$V_1 \; V_4 = W_{14}$

normalização
(soma = 1)

W_{11}

W_{12}

W_{13}

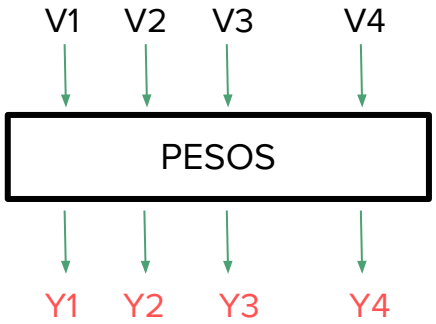
W_{14}

$$W_{11}V_1 + W_{12}V_2 + W_{13}V_3 + W_{14}V_4 = Y_1$$

Recalculamos todos os vetores em função de V_1!

Self Attention

Manga da minha **camiseta**



$W_{11}V_1 + W_{12}V_2 + W_{13}V_3 + W_{14}V_4 = Y_1$

Recalculamos todos os vetores em função de V_1!

$V_1 \; V_1 = W_{11}$

$V_1 \; V_2 = W_{12}$

$V_1 \; V_3 = W_{13}$

$V_1 \; V_4 = W_{14}$

normalização (soma = 1)

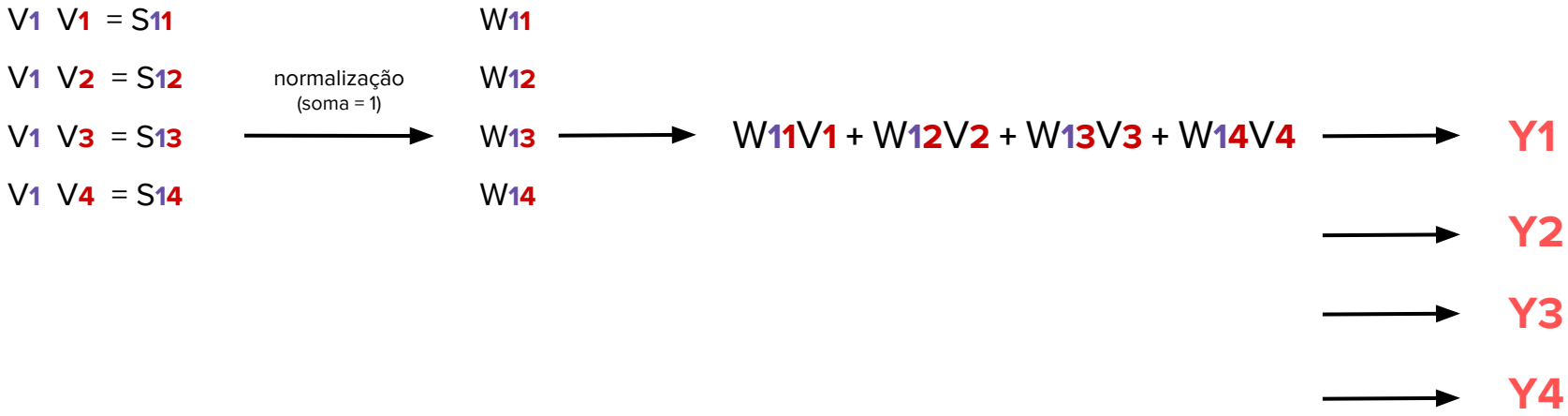
W_{11}

W_{12}

W_{13}

W_{14}

Keys, values, queries



Keys, values, queries

$V_1 \cdot V_1 = S_{11}$

$V_1 \cdot V_2 = S_{12}$

$V_1 \cdot V_3 = S_{13}$

$V_1 \cdot V_4 = S_{14}$

normalização (soma = 1)

W_{11}

W_{12}

W_{13}

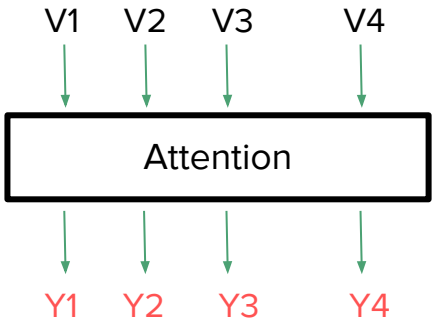
W_{14}

$W_{11}V_1 + W_{12}V_2 + W_{13}V_3 + W_{14}V_4$ → Y_1

→ Y_2

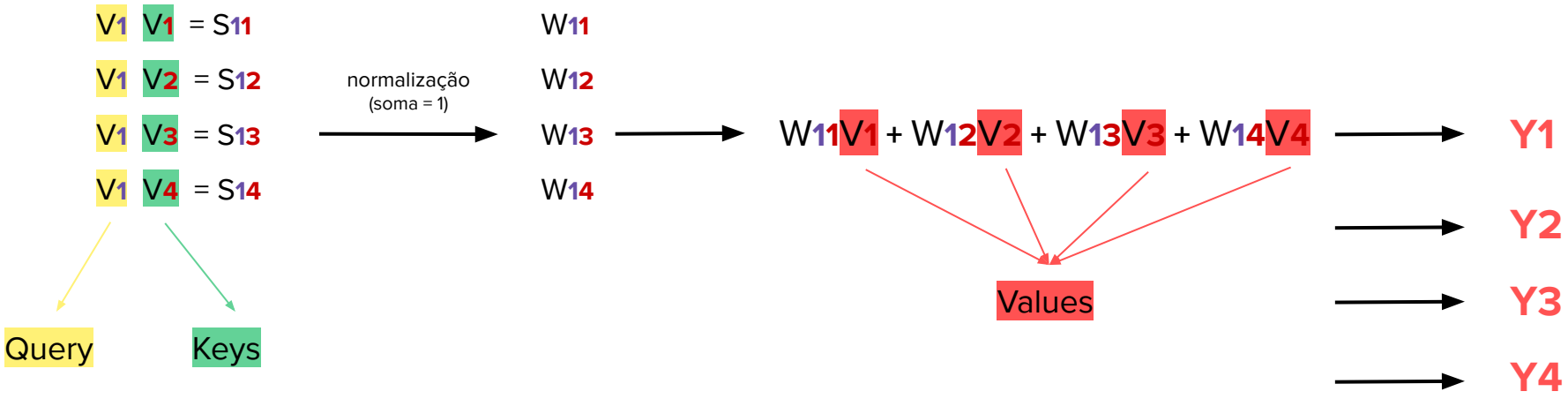
→ Y_3

→ Y_4

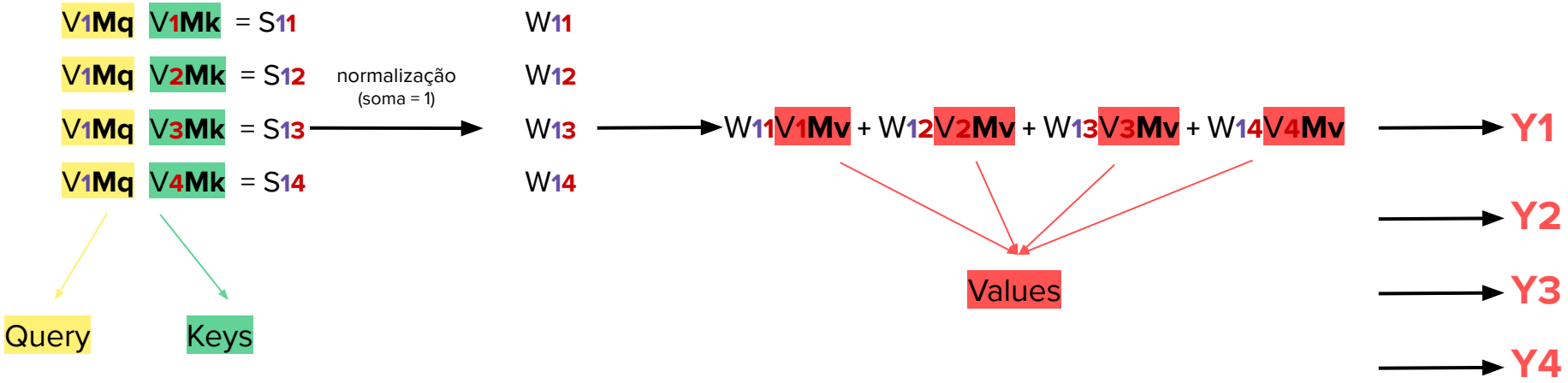


Até agora não temos parâmetros que são atualizados com o treinamento

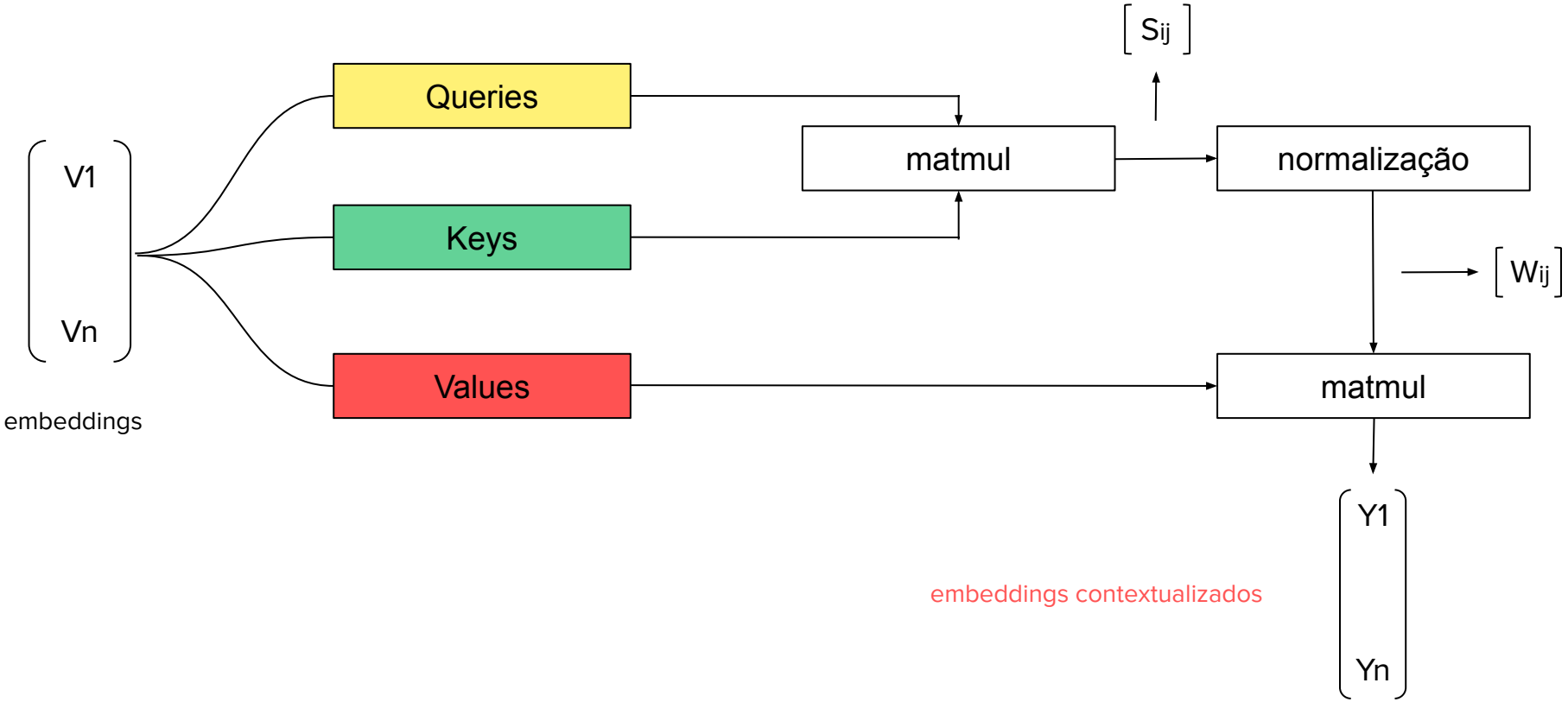
Keys, values, queries



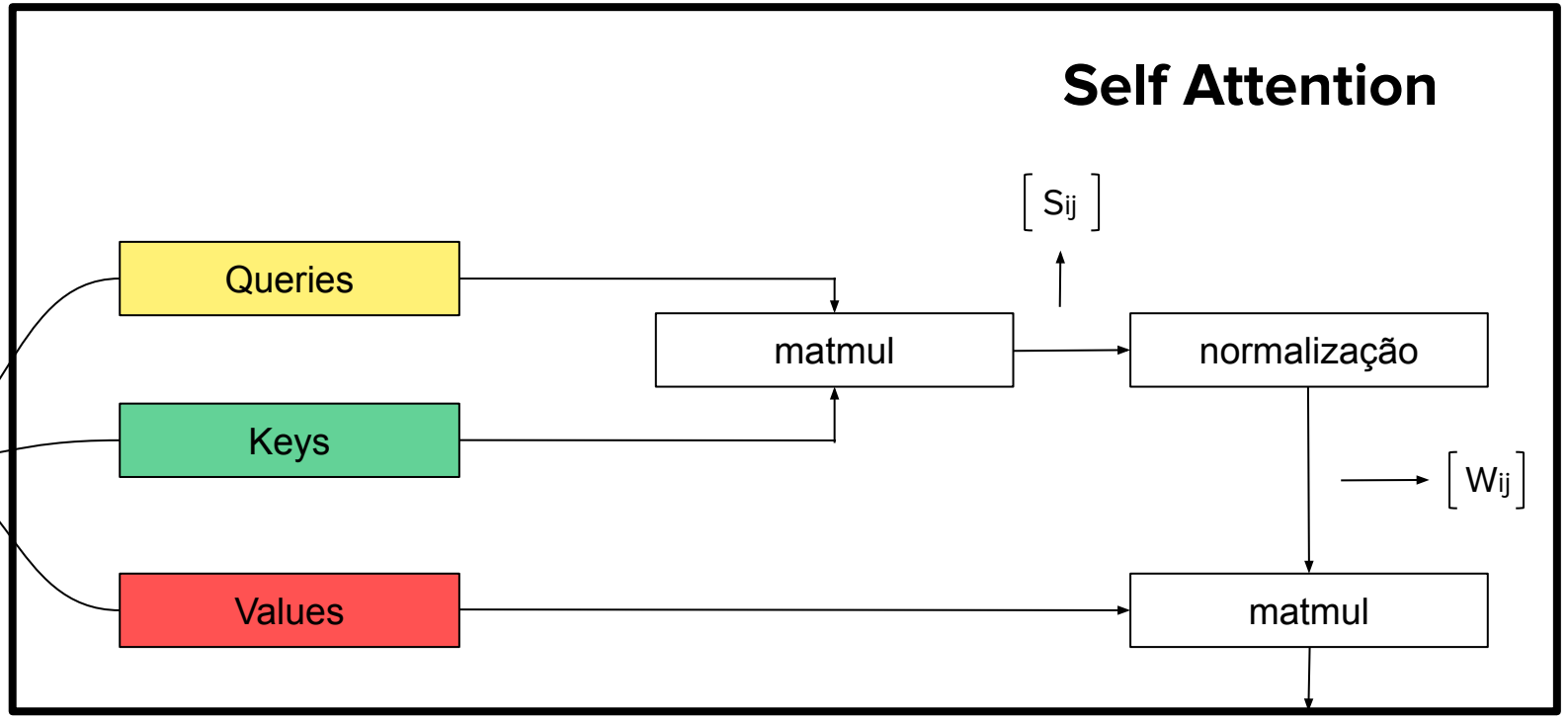
Keys, values, queries



Self Attention



Self Attention



embeddings

embeddings contextualizados

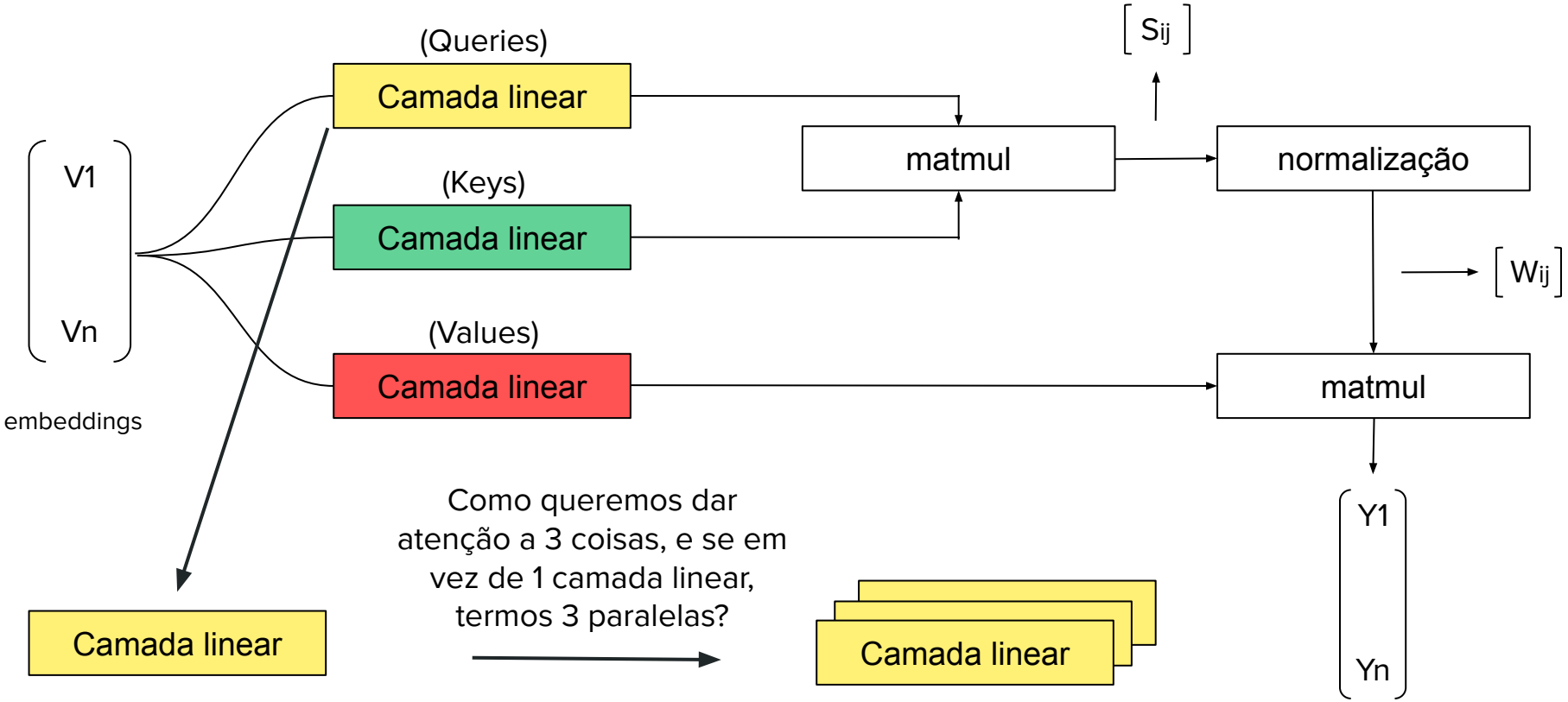
Multi Head Attention

Multi-Head Attention - Introdução

Vamos ver mais um exemplo. Em relação a “deu”, para quais outras palavras queremos dar atenção?

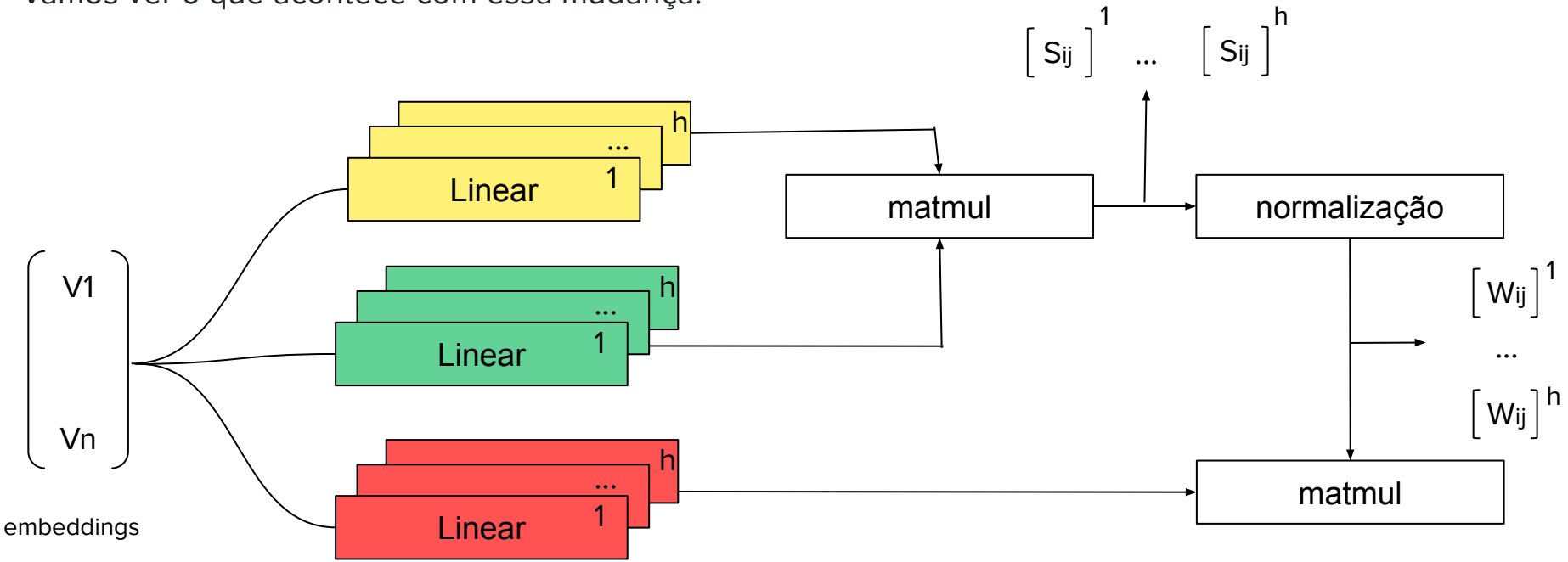
Será que temos atenção suficiente no nosso modelo?

O que temos até agora é...



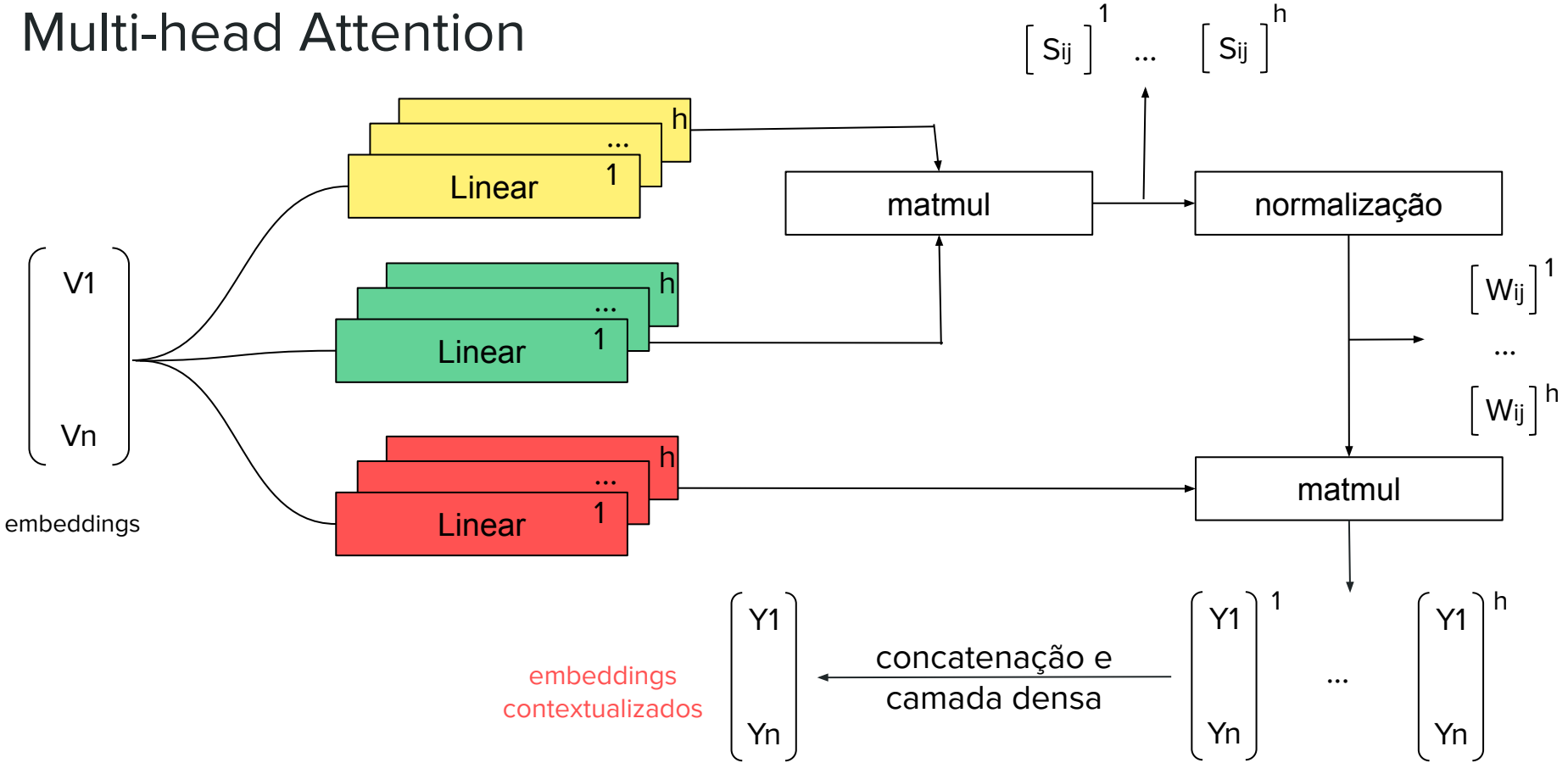
Multi-head Attention

Vamos ver o que acontece com essa mudança:

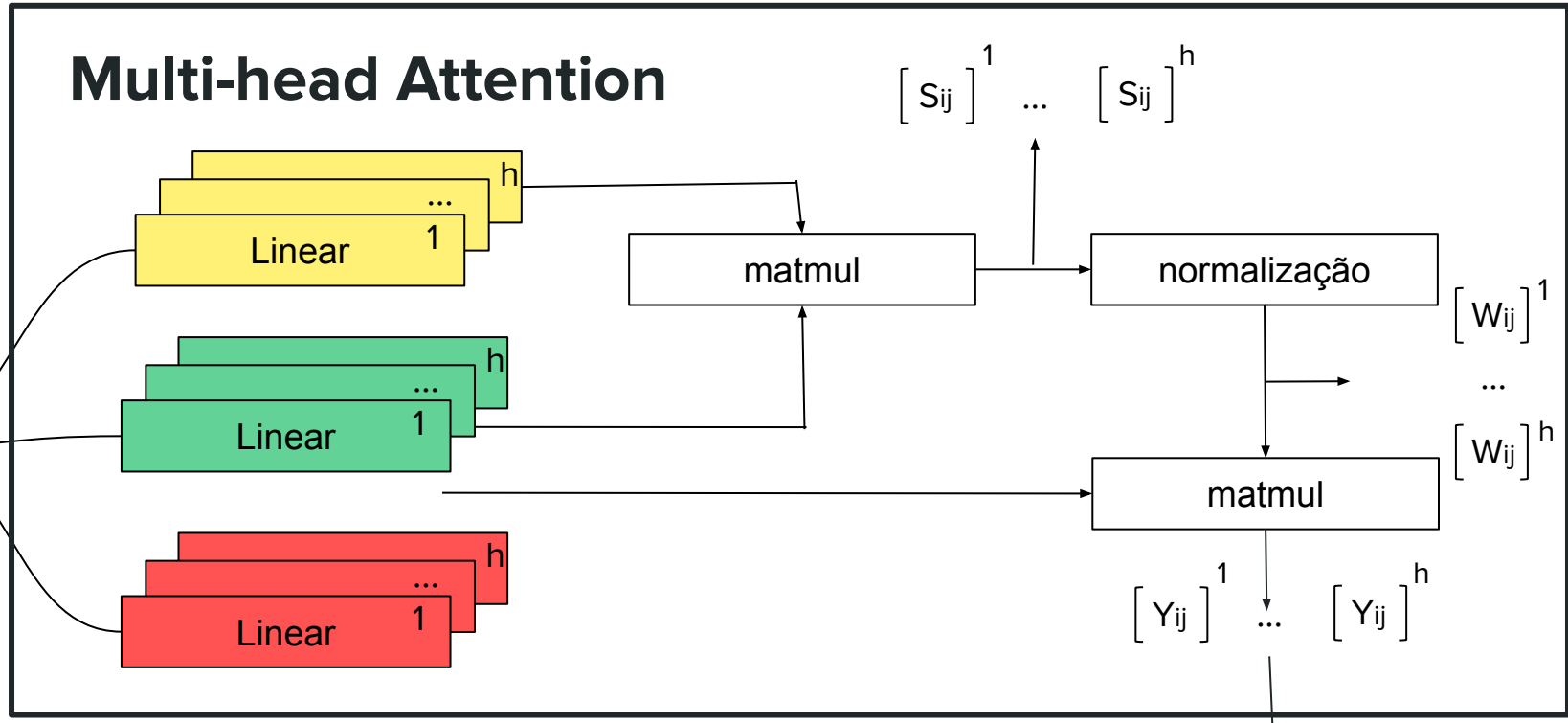


h camadas lineares paralelas, inicializadas com pesos aleatórios diferentes

Multi-head Attention



Multi-head Attention



Obs: o h é o número de ‘heads’ e por isso chamamos de Multi-head Attention. Interpretamos cada head como uma dimensão de significado, como as dimensões dos embeddings.

Transformer

Um pouco de contexto

- Antes dos transformers, a arquitetura mais utilizada era a de RNNs
- As RNNs utilizam um mecanismo de recorrência, ou seja, uma frase precisa ser processada sequencialmente, uma palavra por vez
- No contexto de seq2seq, foi observado que layers de atenção melhoraram muito a performance das redes
    - Ou seja, as melhores soluções (para tradução especialmente) utilizavam tanto recorrência como atenção

A arquitetura transformer

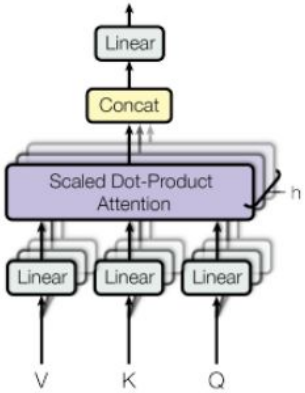
- No final de 2017 foi publicado o paper “Attention is all you need”, como o próprio nome diz, foi proposta uma arquitetura de rede neural em que a recorrência era “jogada fora” ficando apenas com o mecanismo de atenção
- Jogar a recorrência fora não sai de graça - em geral as redes transformers precisam de várias camadas para terem bom desempenho, ao contrário das redes recorrentes
- Nesse tipo de rede, o tipo de camada principal é a vista “multi-headed-attention”

Paralelização - o grande diferencial

- Como as redes transformers não precisam processar as palavras de forma sequencial, como as RNNs, é permitida que as operações sejam paralelizadas.
  - Numa RNN a frase “oi tudo bem?” teria a palavra “oi” processada, e o resultado desse processamento seria input para o processamento de “tudo” e assim em diante
  - Numa rede transformer a frase “oi tudo bem?” é processada inteira ao mesmo tempo
- Essa paralelização permite que seja feito um uso mais efetivo das GPUs
  - GPUs tem muito mais cores do que CPUs
  - A evolução das GPUs está muito mais acelerada do que as das CPUs
- Com a paralelização, as arquiteturas mais e mais profundas se tornaram praticáveis - esses modelos gigantes tem tidos resultados impressionantes

O primeiro transformer

- Encoder-Decoder
- Positional encoding
- Teacher forcing



Multi-Head Attention