(6) Redes neurais para dados sequenciais

Redes Neurais e Arquiteturas Profundas

Moacir A. Ponti
CeMEAI/ICMC, Universidade de São Paulo
MBA em Ciência de Dados

www.icmc.usp.br/~moacir — moacir@icmc.usp.br

São Carlos-SP/Brasil – 2020

Agenda

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

Agenda

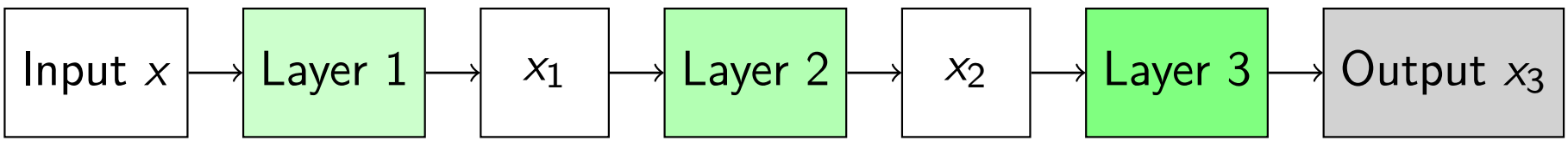
Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

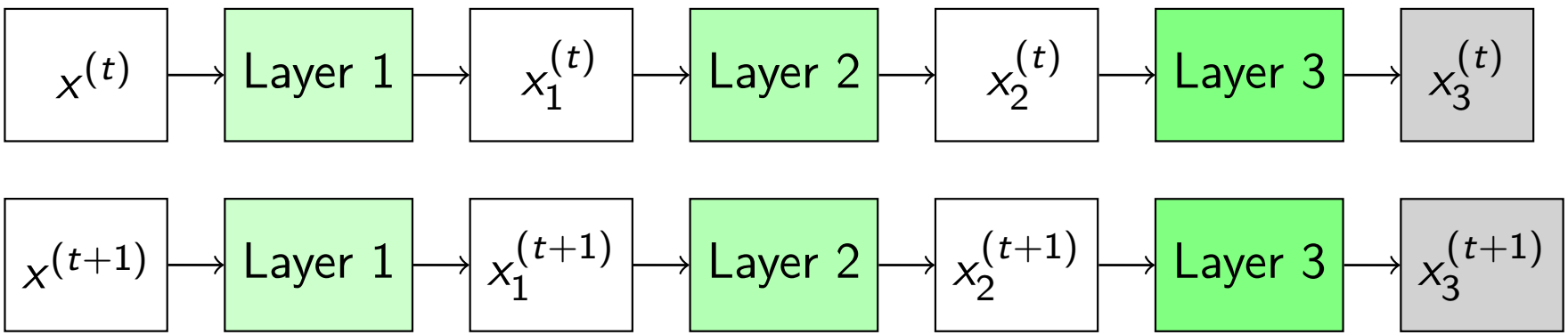
Para dados não sequenciais

- Camadas densas e convolucionais consideram apenas o exemplo atual para computar a saída
- Em cada iteração, cada entrada vai passando pelas camadas até atingir a saída



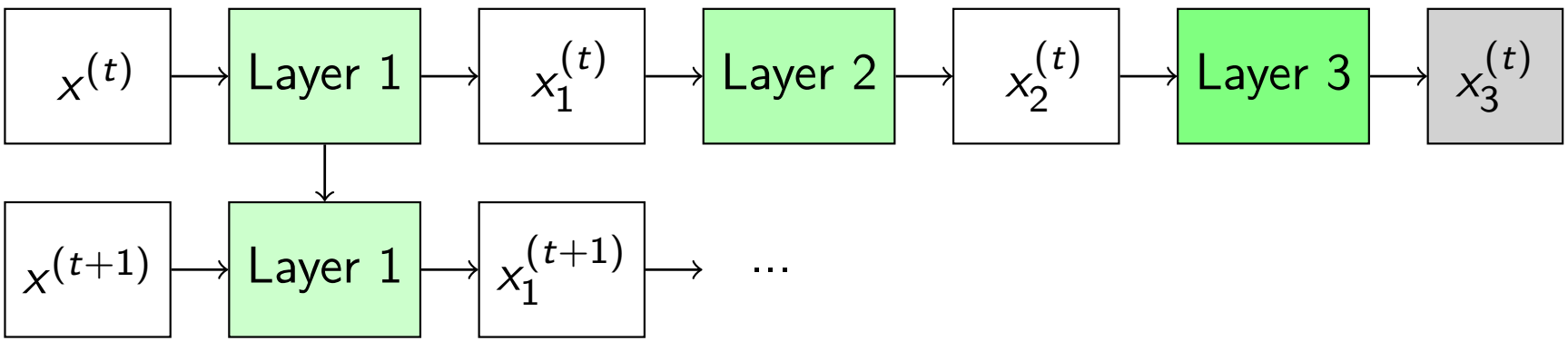
Para dados não sequenciais

- Na iteração $t+1$ usamos os dados de $t+1$ para adaptar os parâmetros



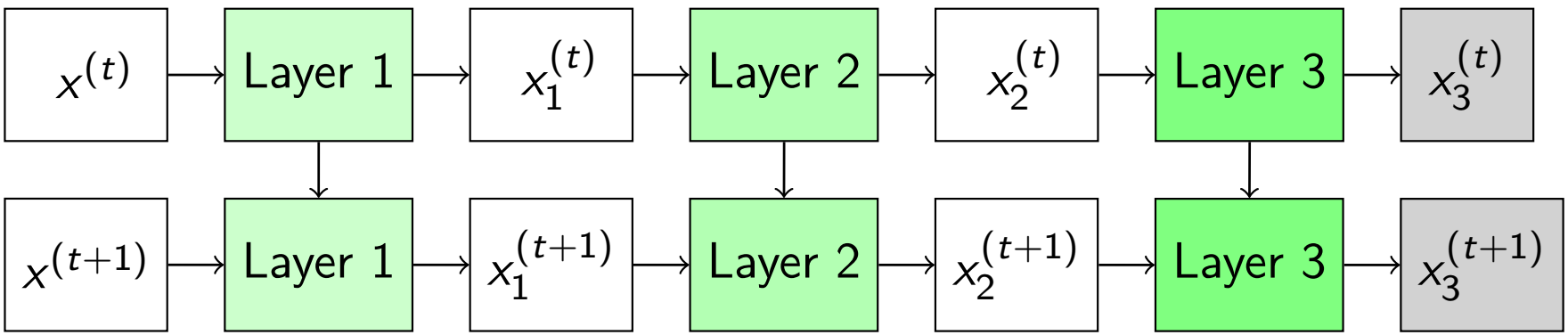
Para dados sequenciais

- Se a iteração $t+1$ depende da anterior t, usamos a saída de cada camada para alimentar a camada na entrada da iteração $t+1$



Para dados sequenciais

- Dessa forma, a saída (após a primeira), dependerá não apenas da entrada atual, mas das saídas computadas anteriormente para cada unidade

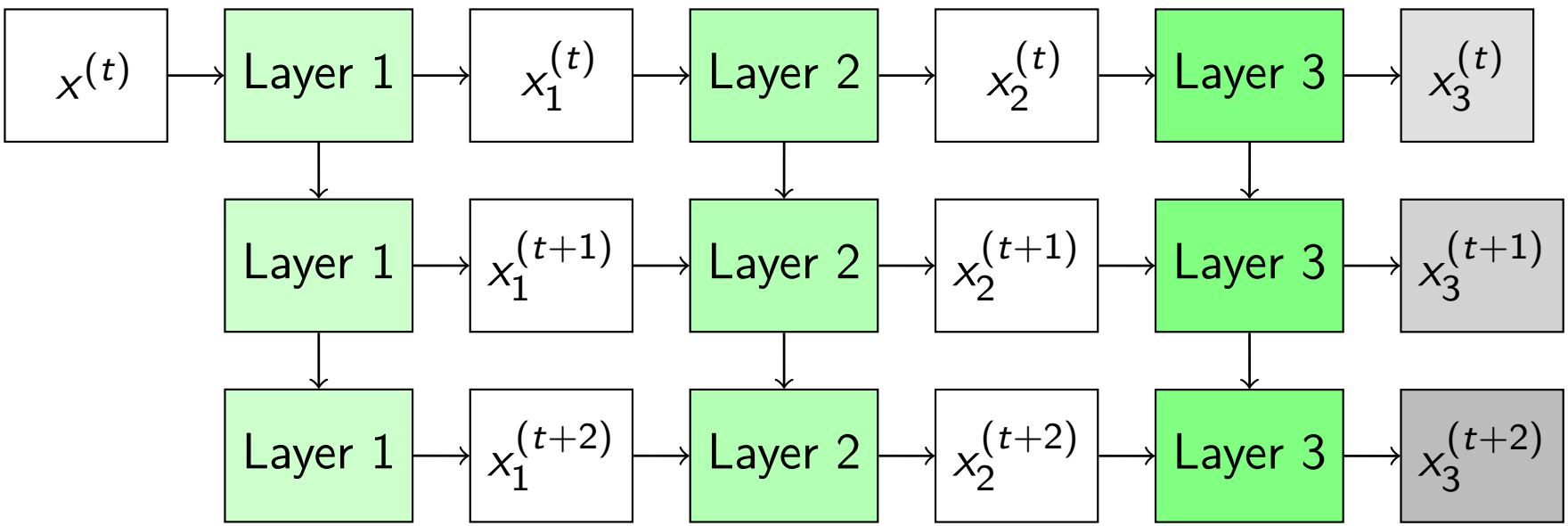


Configurações de sequências

- Uma entrada, saída sequencial
- Entrada sequencial, uma saída
- Entrada sequencial, saída sequencial

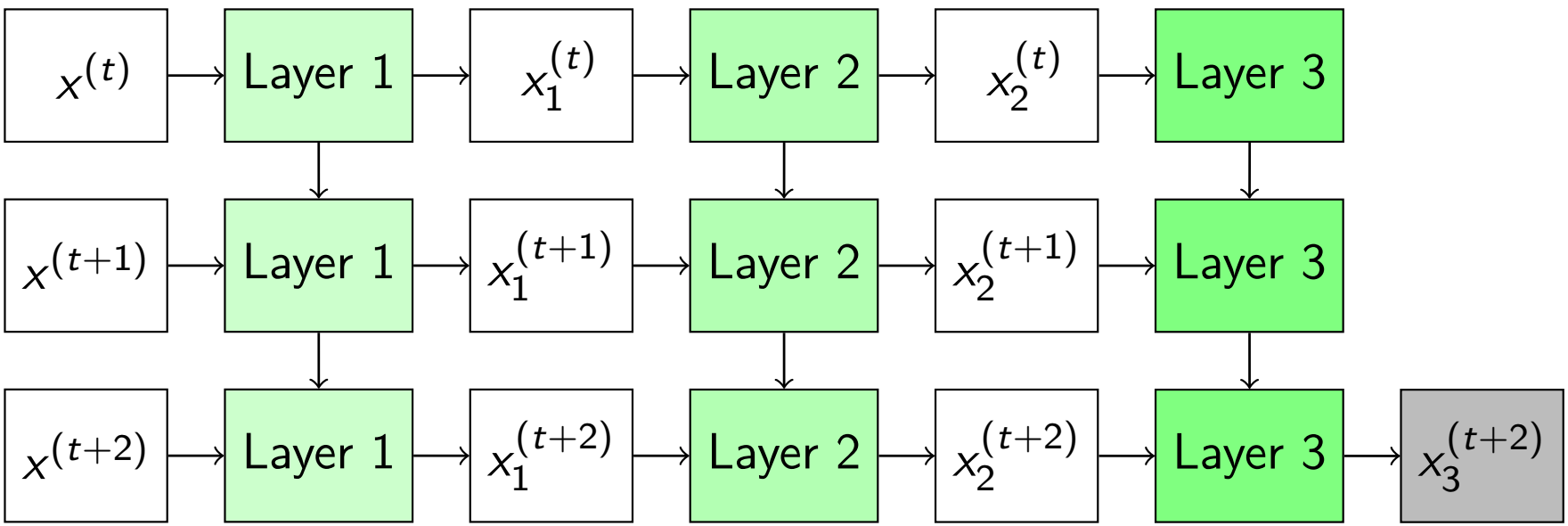
Configurações de sequências

- Uma entrada, saída sequencial: e.g. um áudio ou imagem é dado como entrada e a rede produz uma sequência de palavras que os descrevem



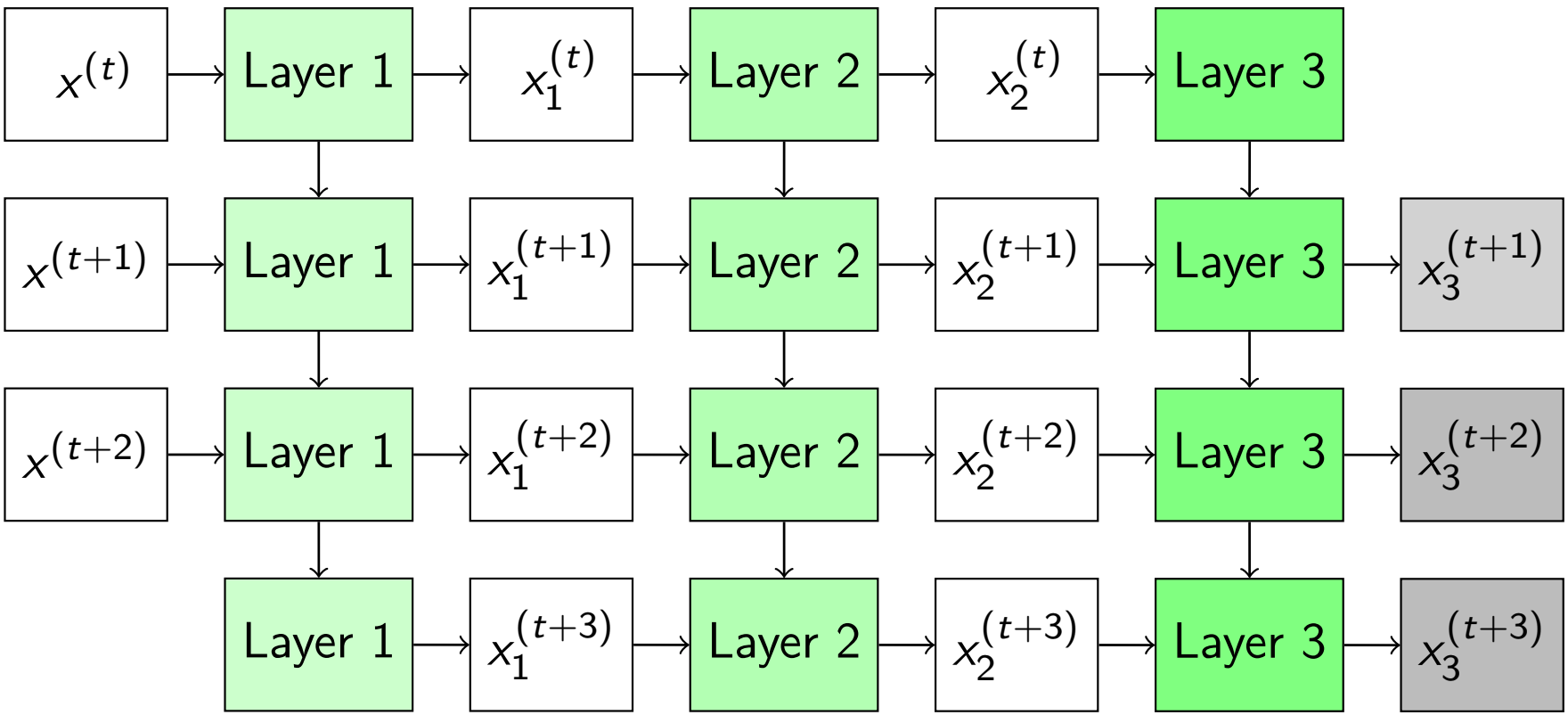
Configurações de sequências

- Entrada sequencial, uma saída e.g. um texto é dado como entrada e a saída é sua análise de sentimentos: conteúdo positivo ou negativo.



Configurações de sequências

- Entrada sequencial, saída sequencial e.g. tradução automática de sentenças entre diferentes idiomas (com atraso k)



Agenda

Dados sequenciais: recorrência

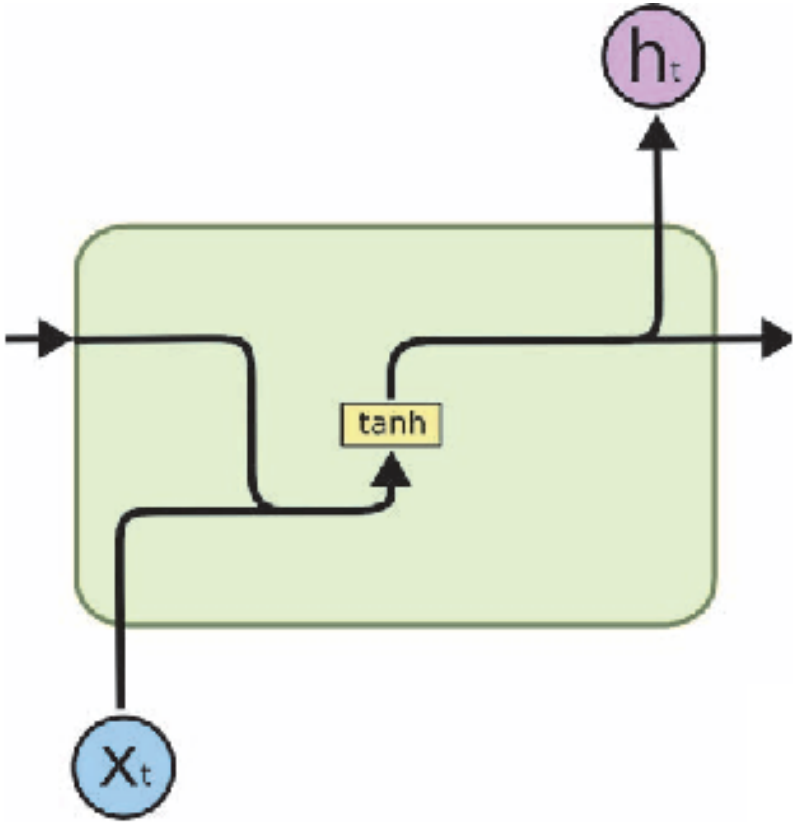
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

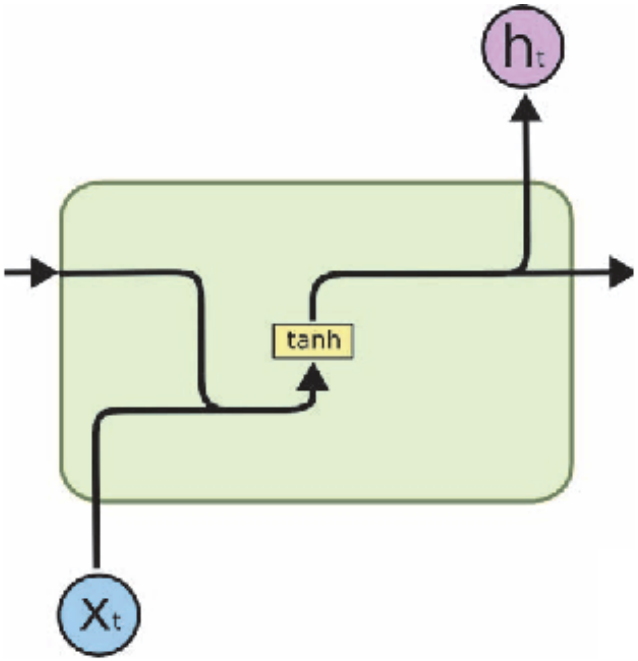
Sequence-to-Sequence e Mecanismo de Atenção

- Transformer Network

Aprende um tipo de "memória"



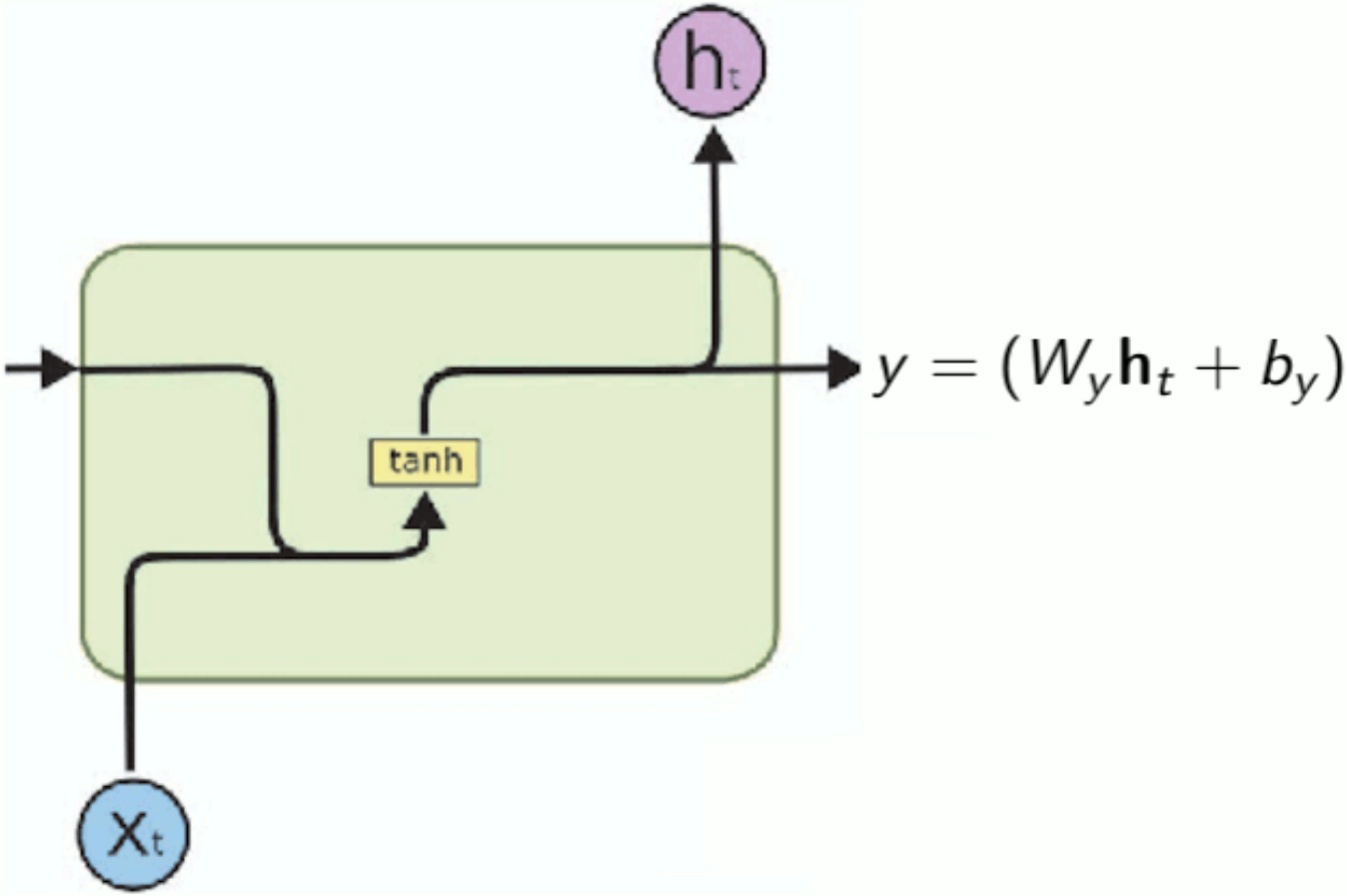
Componentes são combinações lineares



$$h_t = \tanh\left(W_h h_{t-1} + W_x x_t + b_h\right)$$
$$y = \left(W_y h_t + b_y\right)$$

Saída recorrente (sumário) e saída da rede

$$\mathbf{h}_t = \tanh\left(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + b_h\right)$$



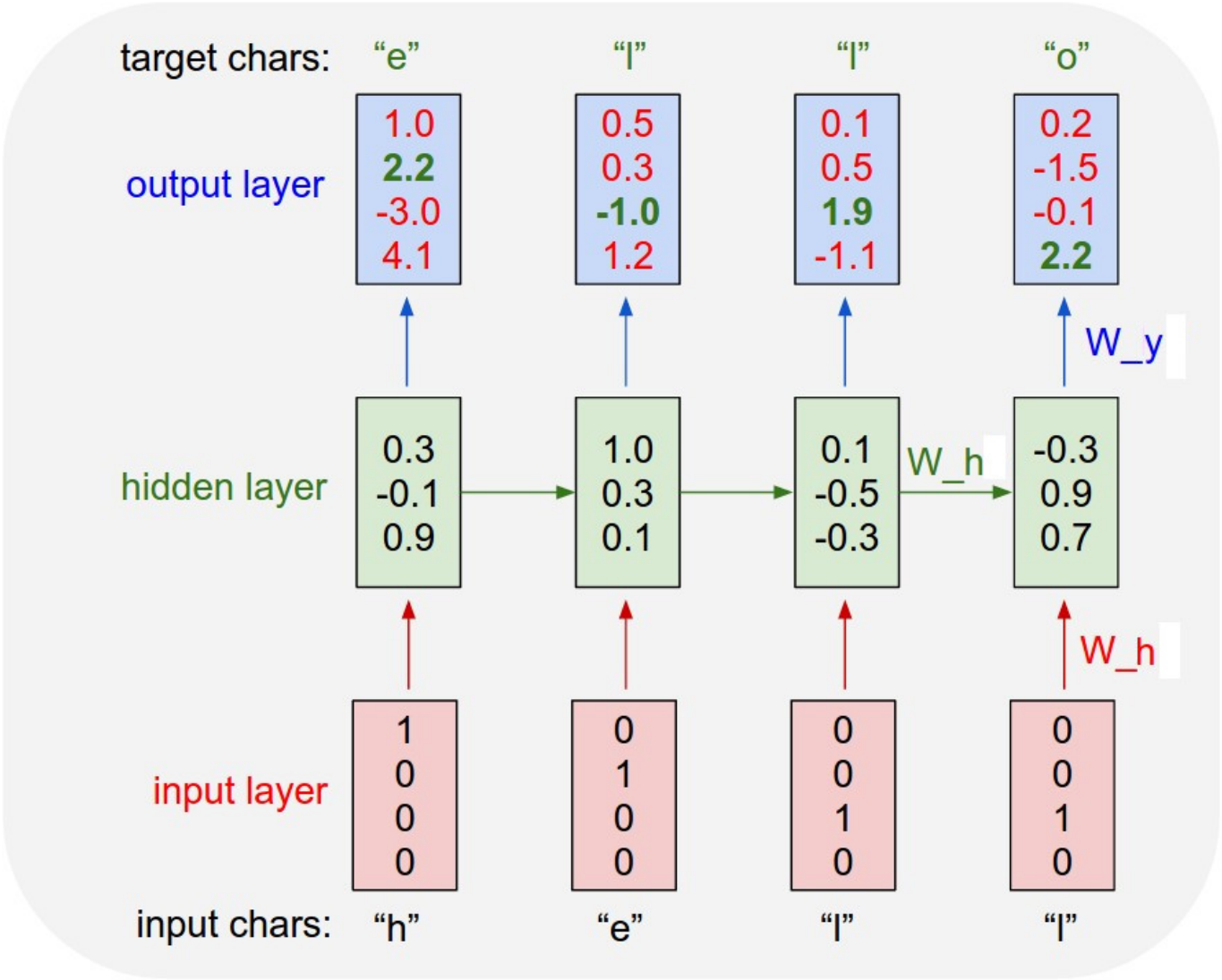
Exemplo: predizer próximo caracter

Definimos uma codificação one-hot para os caracteres:

- `h` $= [1, 0, 0, 0]$
- `e` $= [0, 1, 0, 0]$
- `l` $= [0, 0, 1, 0]$
- `o` $= [0, 0, 0, 1]$

`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

Exemplo: predizer próximo caracter



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

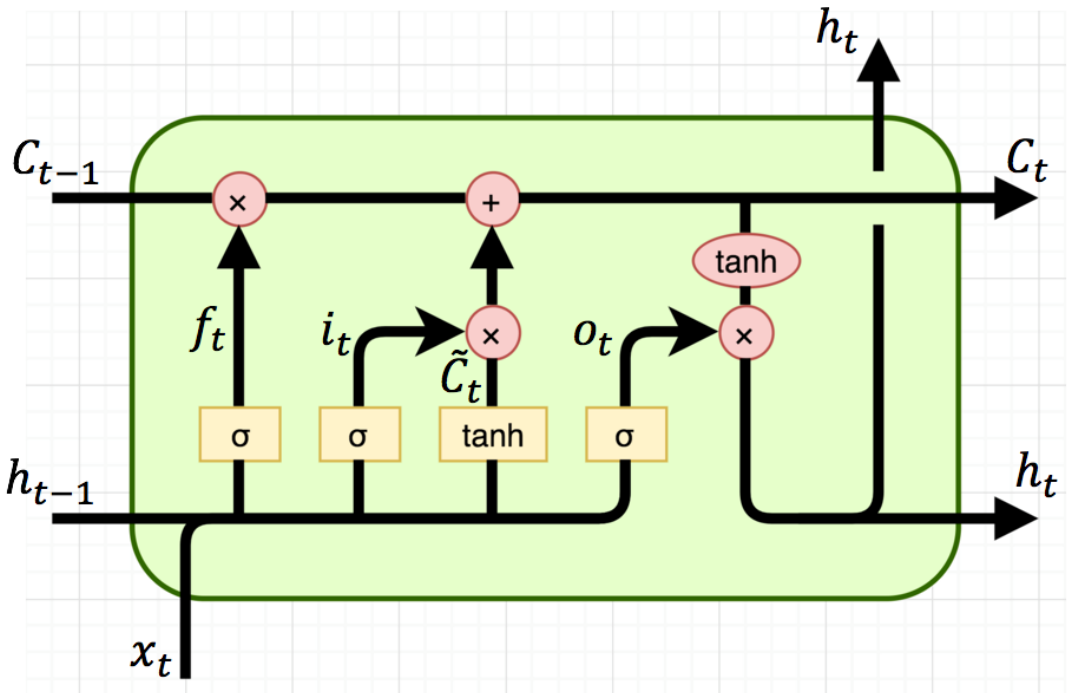
Agenda

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

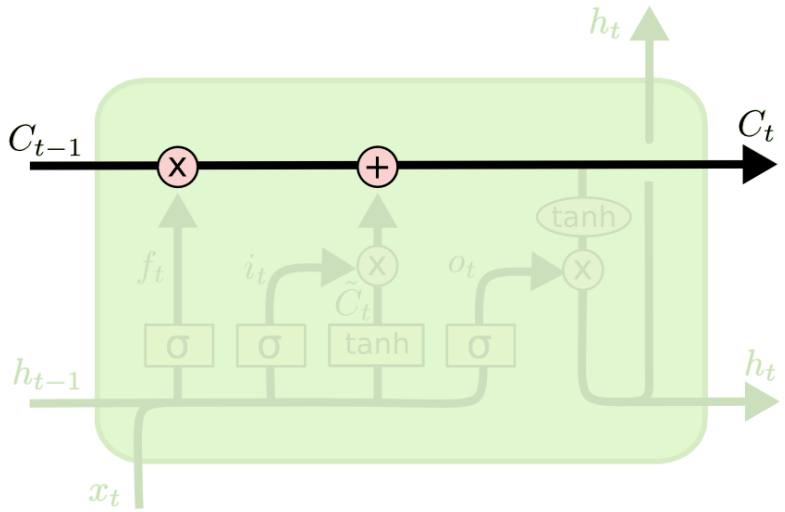
Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

Long Short Term Memory Unit (LSTM)



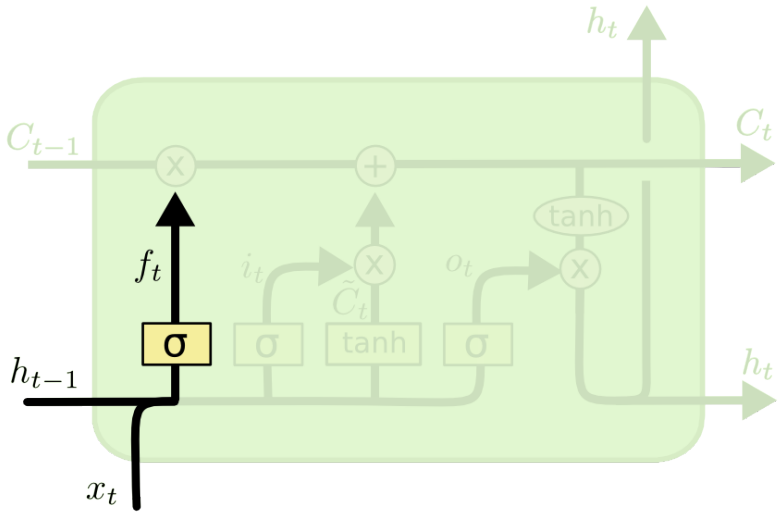
Adaptados de `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

LSTM network: Cell state



- Responsável pela memória longa da unidade, adiciona contribuições para além da iteração anterior.
- Esse estado pode ser modificado por 2 portões/**gates**

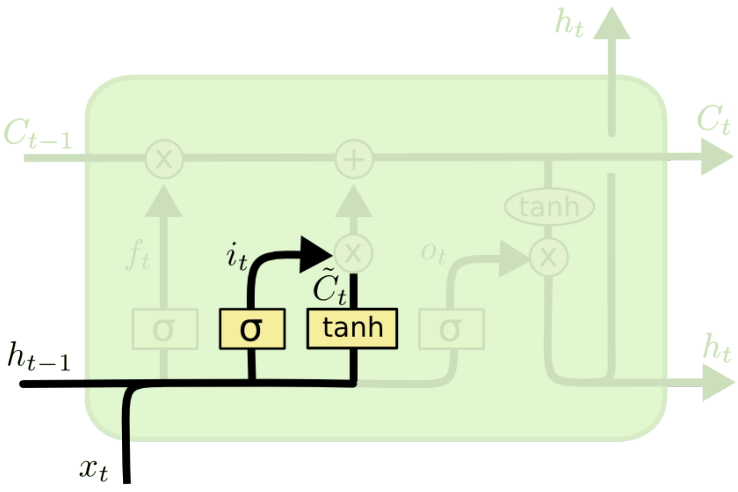
LSTM network: forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- decide o que cancelar de C com base no sumário anterior e a entrada atual
- saída entre 0 (esquecer) e 1 (manter totalmente) para cada dimensão de C

LSTM network: input gate

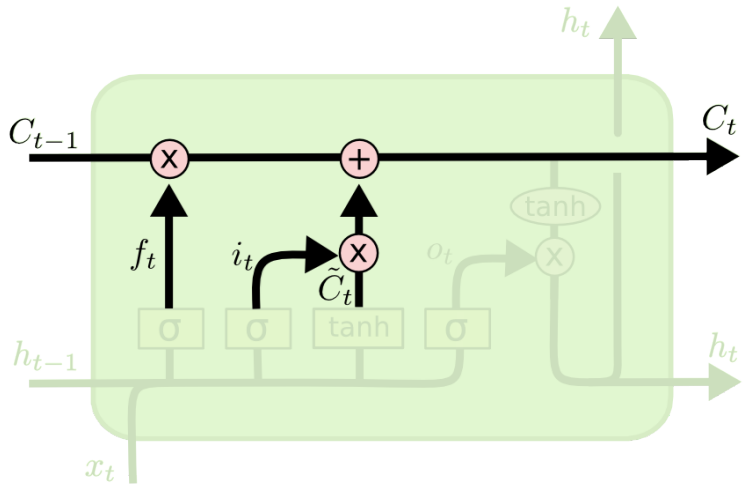


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- primeiro, combina o sumário anterior h_{t-1} e a entrada x_t
- então, aprende um filtro $\tilde{C}_t$ que indica quais partes devem ser mantidas na "memória longa", sendo depois somado a C_{t-1}

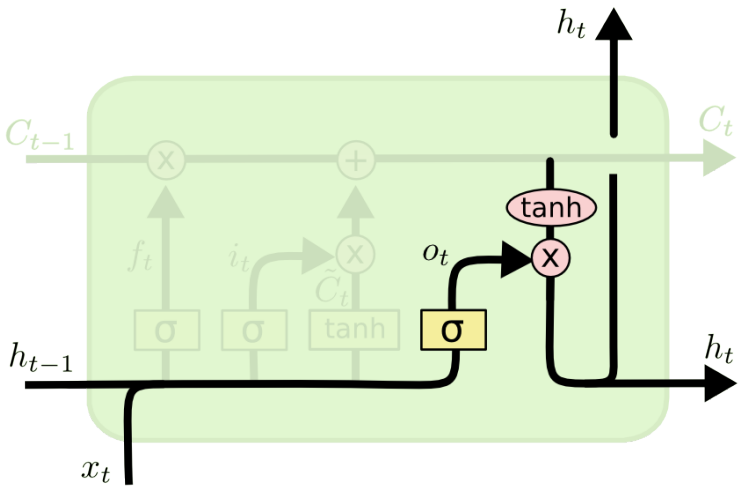
LSTM network: update Cell state



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- agora temos uma combinação entre os estados atual e anterior
- acima o $*$ significa uma multiplicação ponto-a-ponto

LSTM network: output gate



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- decide qual será o sumário, transformado a partir do anterior
- e ponderado de acordo com o estado de célula atual, C_t

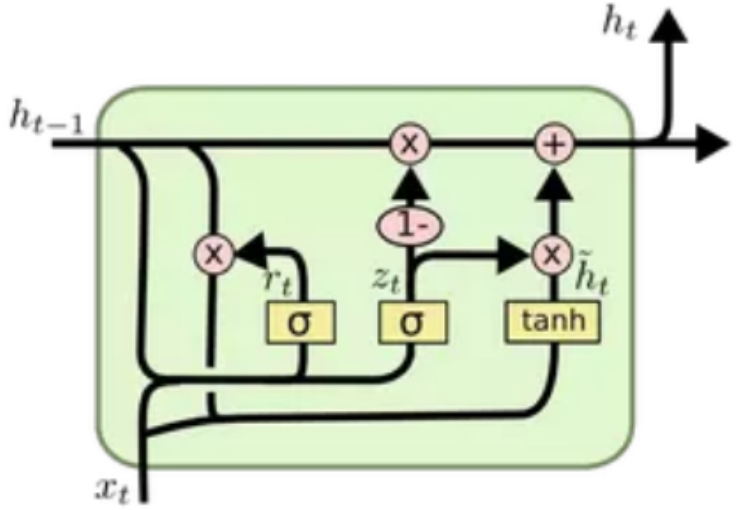
Agenda

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- Não possui Cell State
- Reset gate r: filtra qual parte de h_{t-1} será utilizada para compor o novo sumário candidato em conjunto com x_t
- Update gate z: pondera partes do sumário anterior de forma complementar ao novo estado candidato
- $\tilde{h}_t$ é o sumário "candidato"

GRU x LSTM

- Não há um consenso de "melhor" método
- GRU em muitos casos tem resultados similares à LSTM, com menos parâmetros
- Há uma versão recente, JANET, que simplificou ainda mais o modelo, removendo o "Reset gate"
- Temporal Convolutional Networks, que utilizam convoluções 1D para aprender posicionamento local, também se mostraram eficientes em alguns cenários

Agenda

- Dados sequenciais: recorrência
  - Camada recorrente básica (RNN)
  - Long Short Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
- Word2Vec: representações para texto
- Sequence-to-Sequence e Mecanismo de Atenção
  - Transformer Network

Word2Vec

- Representação (embedding) para palavras
- A função de custo para aprender essa representação:

$$p(w_{t+j}|w_t)$$

  t é a palavra, $t + j$ são todas as palavras no contexto de t
- Otimiza em função de palavras que devem estar próximas se estiverem no mesmo contexto

Word2Vec

- Representação (embedding) para palavras
- A função de custo para aprender essa representação:

$$p(w_{t+j}|w_t)$$

  t é a palavra, $t + j$ são todas as palavras no contexto de t
- Otimiza em função de palavras que devem estar próximas se estiverem no mesmo contexto

Word2Vec: skipgrams

Skip-grams (SG)

Predição de palavras em uma certa "janela" de proximidade m de uma palavra t

- Formulação "softmax":

$$\frac{\exp(u_o^T v_c)}{\sum_w^V \exp(u_w^T v_c)}$$

V é o total de palavras no vocabulário
u_o é a representação de uma palavra de "saída" (Que queremos prever
v_c é a representação de uma palavra de entrada (central)

Word2Vec: skipgram

Dada uma representação one-hot de uma palavra $w_t \in R^V$, calculamos sua representação vetorial $v_c \in R^d$ (central)

$$W \cdot w_t = v_c$$

$$\begin{bmatrix} \cdots & \cdots & 0.1 & \cdots & \cdots \\ \cdots & \cdots & -0.3 & \cdots & \cdots \\ \cdots & \cdots & 1.4 & \cdots & \cdots \\ \cdots & \cdots & 0.2 & \cdots & \cdots \\ \cdots & \cdots & 0.5 & \cdots & \cdots \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.3 \\ 1.4 \\ 0.2 \\ 0.5 \end{bmatrix}$$

Word2Vec: skipgram

v_c é filtrada por representações u_o das palavras de saída (no contexto, que queremos prever) em diferentes posições $t - i$

$$u_o^T \cdot v_c$$

Para todas as palavras do vocabulário isso é codificado em uma matriz:

$$U_o \cdot v_c$$

$$\begin{bmatrix} 0.0 & 2.0 & 0.1 & 2.0 & 0.1 \\ 0.0 & 1.0 & 2.0 & -0.5 & 1.0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.3 \\ 1.4 \\ 0.2 \\ 0.5 \end{bmatrix} = softmax \left(\begin{bmatrix} 00 \\ 2.9 \\ 0.1 \\ 1.4 \\ -0.5 \\ 0.0 \\ 0.0 \end{bmatrix} \right) = \begin{bmatrix} 0.04 \\ 0.67 \\ 0.04 \\ 0.15 \\ 0.02 \\ 0.04 \\ 0.04 \end{bmatrix}$$

Word2Vec: skipgram

- W aprende representações (nas colunas) para cada palavra quando são "centrais"
- U_o aprende representações (nas linhas) para cada palavra quando são "contexto"

$$W = \begin{bmatrix} \cdots & \cdots & 0.1 & \cdots & \cdots \\ \cdots & \cdots & -0.3 & \cdots & \cdots \\ \cdots & \cdots & 1.4 & \cdots & \cdots \\ \cdots & \cdots & 0.2 & \cdots & \cdots \\ \cdots & \cdots & 0.5 & \cdots & \cdots \end{bmatrix} \quad U_o = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ 0.0 & 1.0 & 2.0 & -0.5 & 1.0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

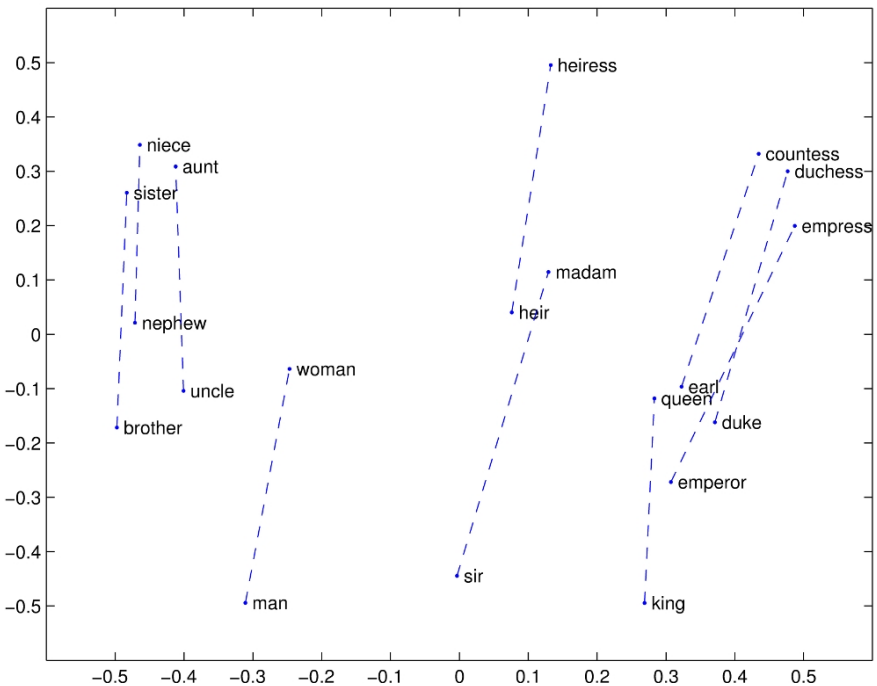
Word2Vec: skipgram

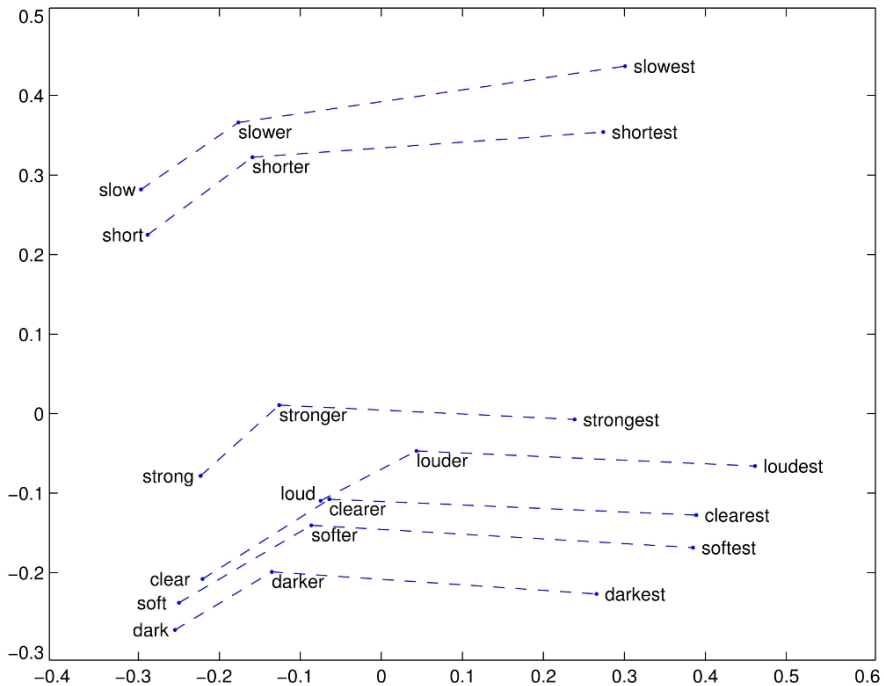
- Palavras que aparecem num mesmo contexto terão representações similares

Deixa o menino jogar
Deixa o moleque jogar
Deixa o piá jogar
Deixa seu filho jogar

Word2Vec: GloVe (Global Vectors for Word Representation)

https://nlp.stanford.edu/projects/glove/





Word Embeddings em Português - NILC/ICMC

http://www.nilc.icmc.usp.br/embeddings

Agenda

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

RNNs e Sequence-to-Sequence (seq2seq)

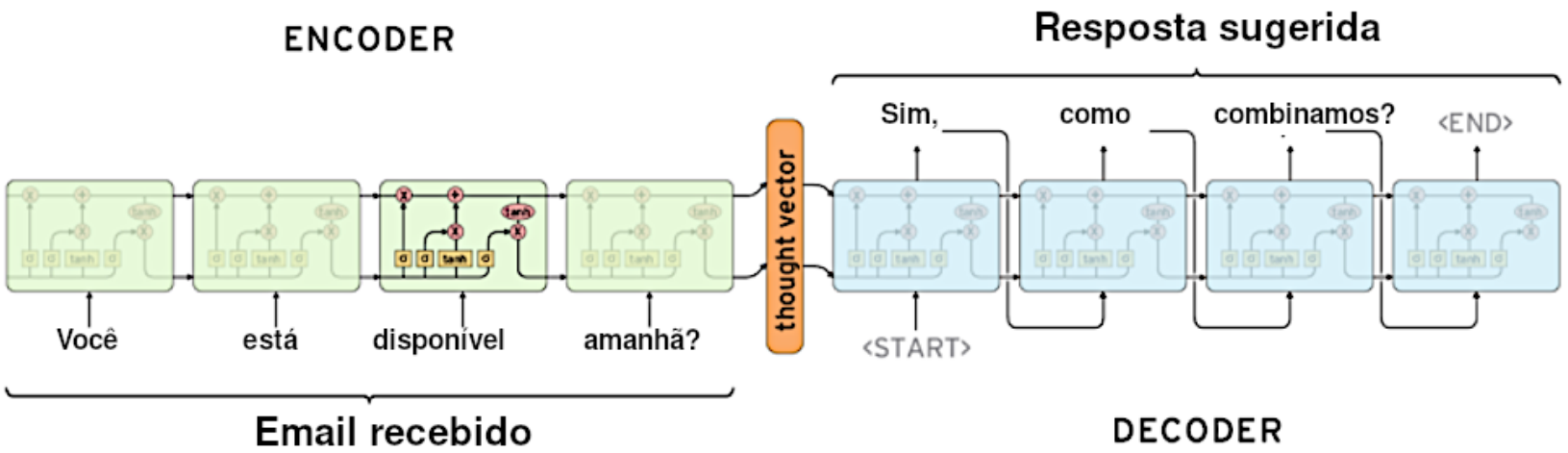


Figura adaptada de: Sachin Abeywardana

- (3 vídeos de Jay Alammar)

Mecanismo de atenção: intuição e motivação

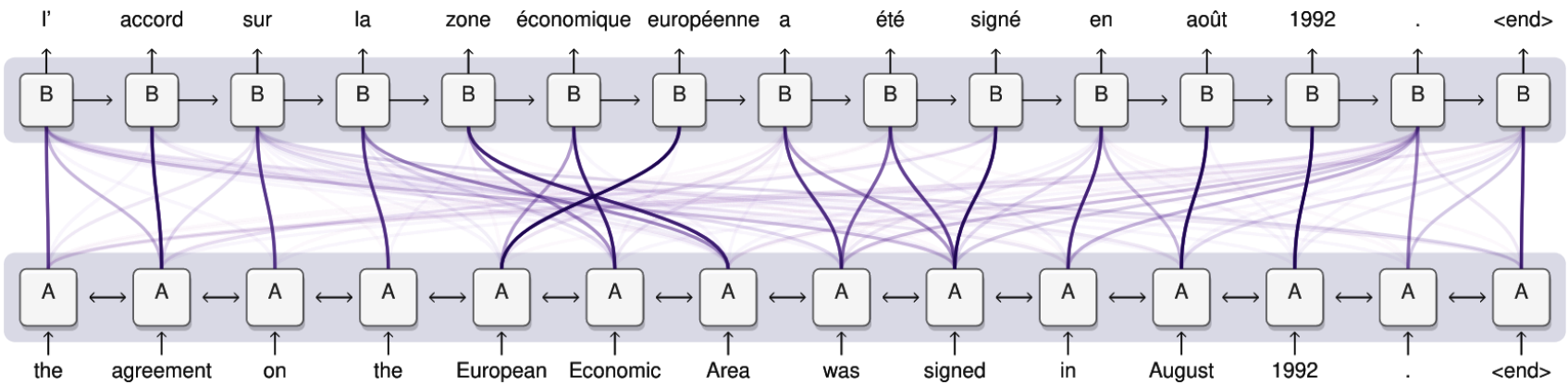
- Encontrar qual parte de uma sequência é mais importante para predizer uma certa saída
- Em unidades recorrentes, cada entrada perturba a memória prejudicando conhecimento de dados anteriores

Mecanismo de Atenção: imagens

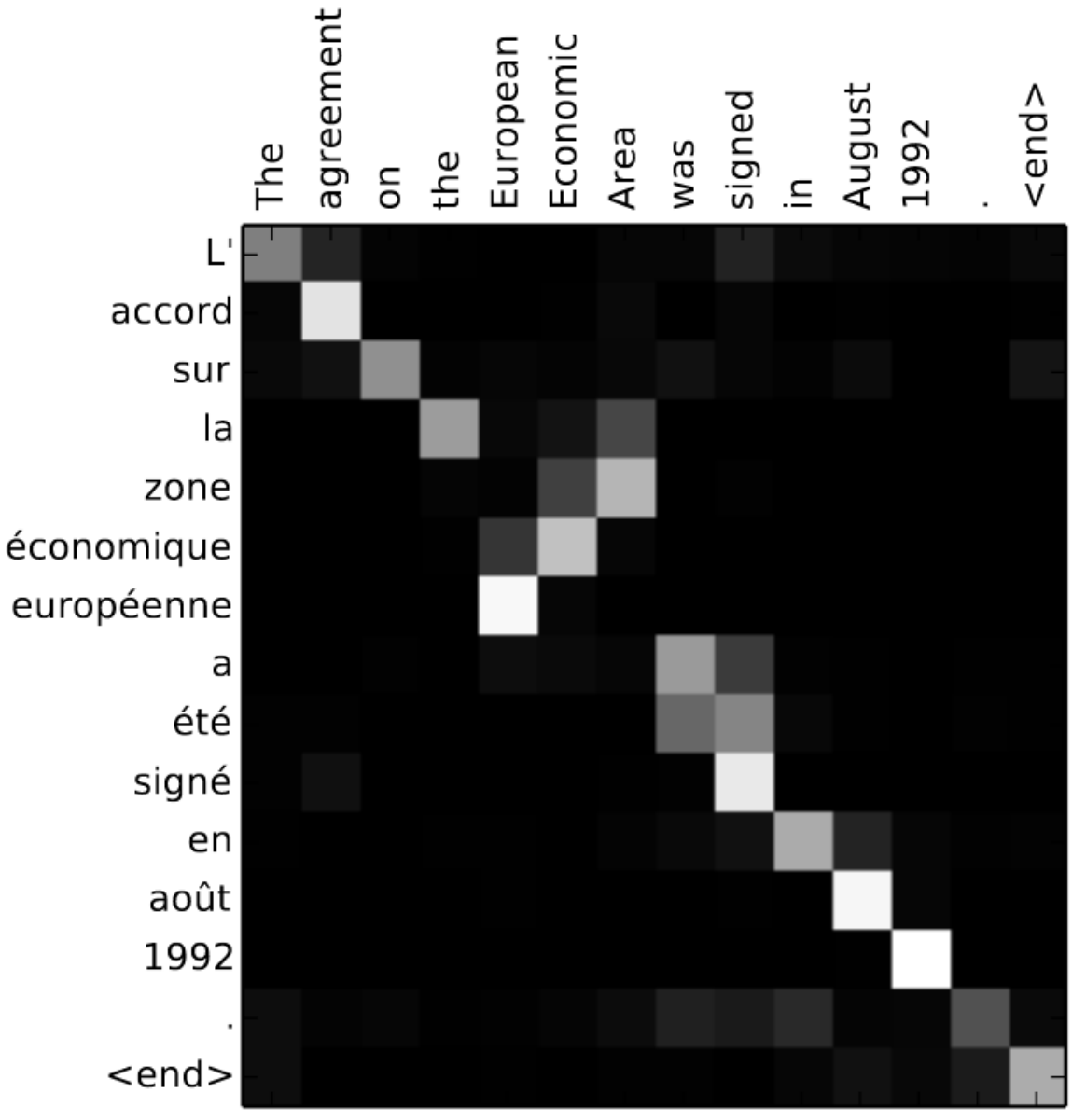
A man wearing a hat and
a hat on a skateboard.

A man is talking on his cell phone
while another man watches.

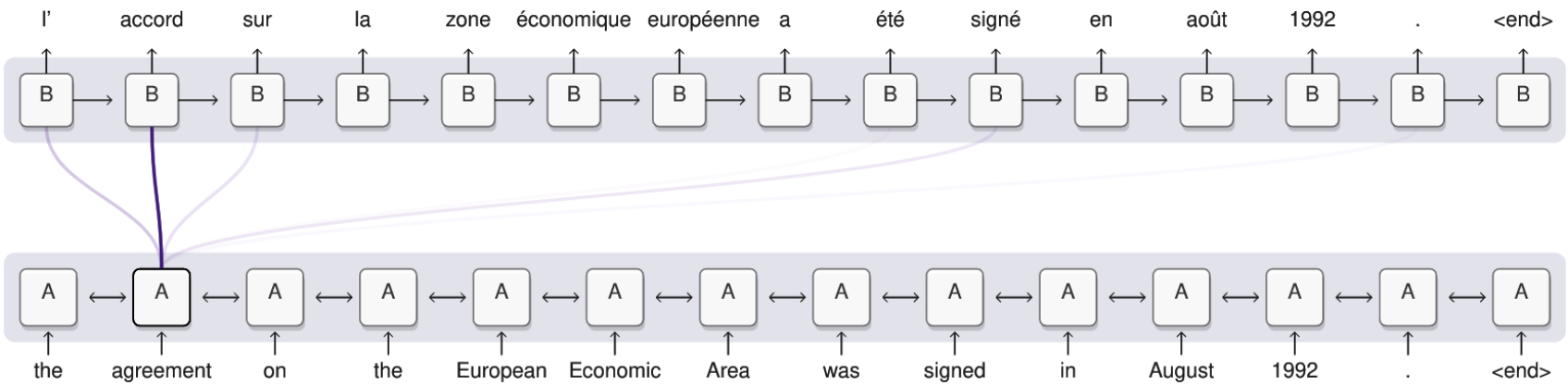
Mecanismo de Atenção: texto



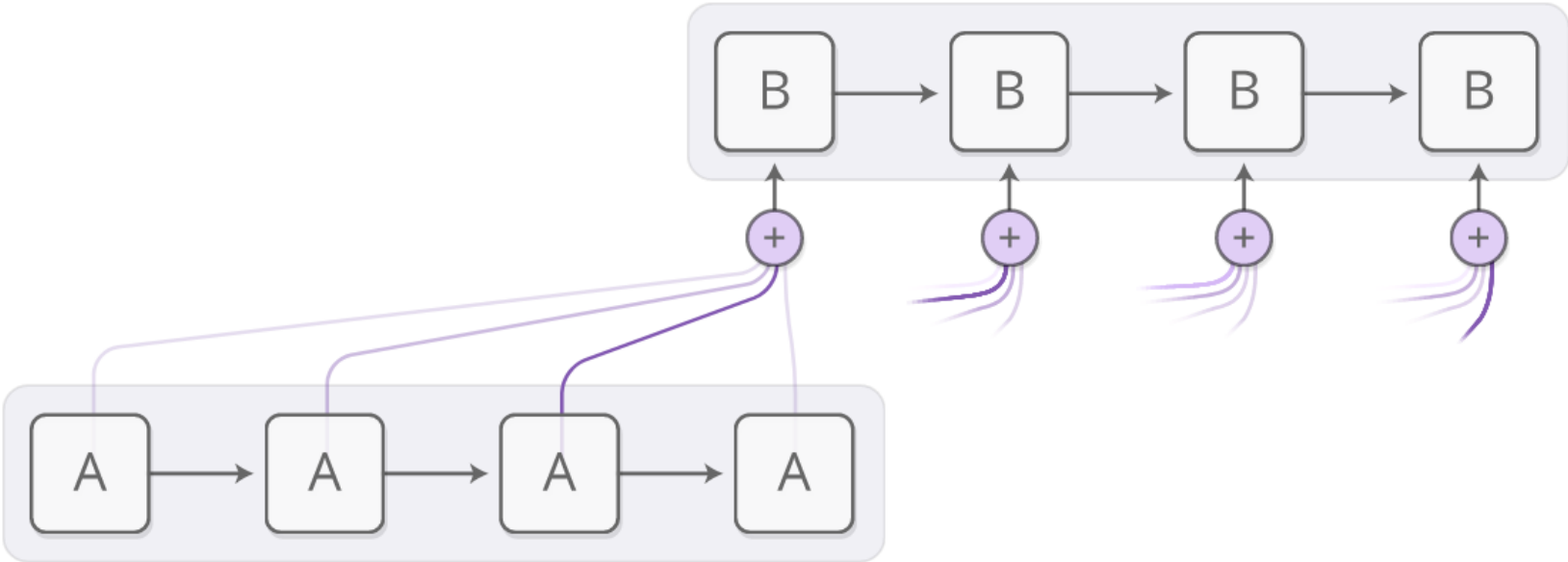
Mecanismo de Atenção: texto



Mecanismo de Atenção: texto



RNNs seq2seq e atenção global



Adaptado de Olah & Carter, "Attention and Augmented Recurrent Neural Networks", Distill, 2016.

http://doi.org/10.23915/distill.00001

- (vídeos de Jay Alammar)

Mecanismo de atenção: implementação básica

- Computar o alinhamento/similaridade entre o sumário atual do decoder, s_i, com sumários anteriores do encoder, h_j

  Usa softmax para obter pesos na forma de probabilidades

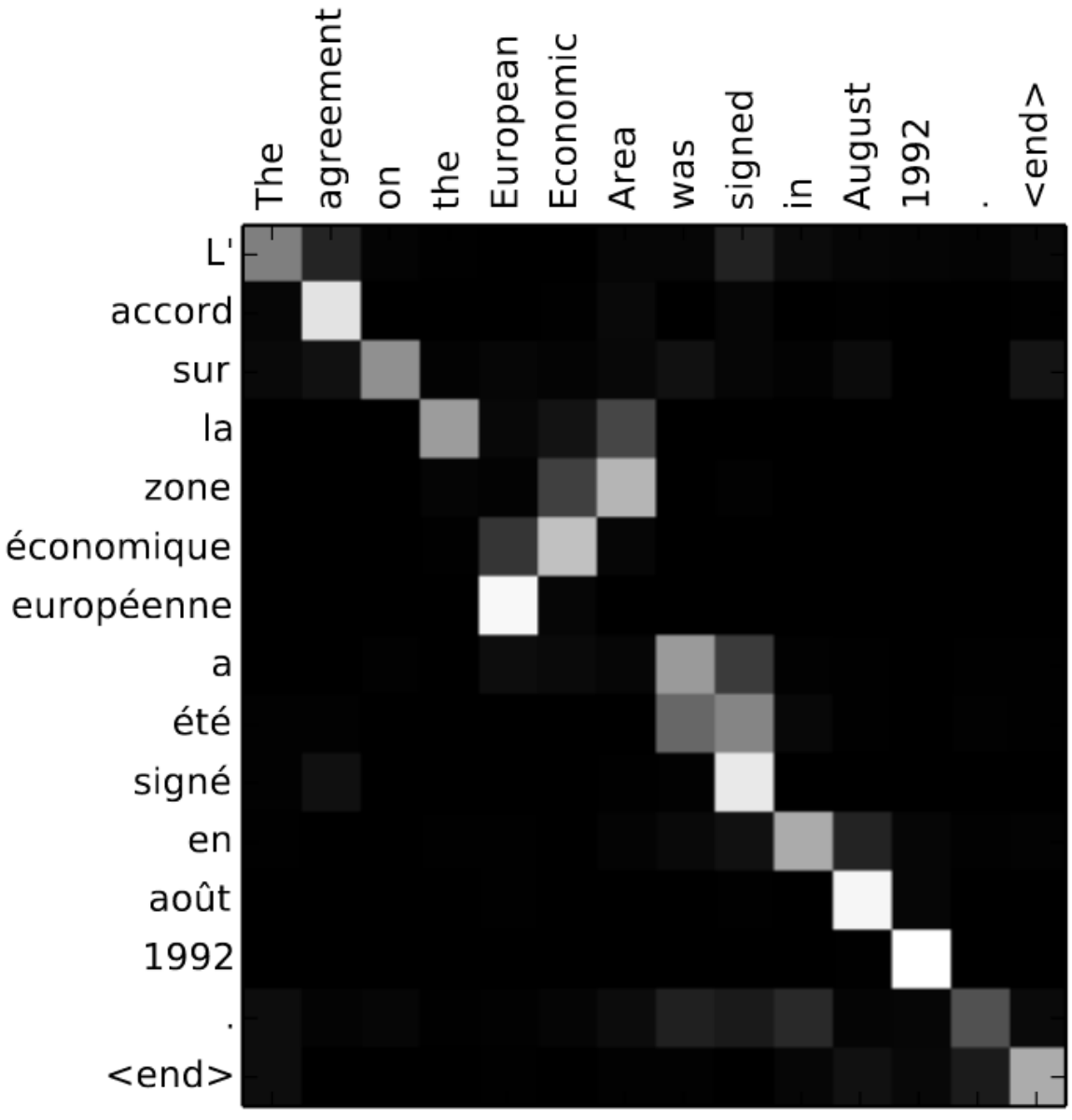
  $$a_{i,j} = \frac{\exp(alinhamento(s_i, h_k))}{\sum_k \exp(alinhamento(s_i, h_k))},$$

  "alinhamento" é um tipo de similaridade, e.g. produto interno:

  $$alinhamento(s_i, h_k)) = s_i^T h_j$$

- Atenção produz um vetor de "contexto" $c_i = \sum_j a_{i,j} h_j$ a ser usado para produzir a saída atual.

Mecanismo de Atenção: exemplo de vetores de alinhamento



Agenda

Dados sequenciais: recorrência
- Camada recorrente básica (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

RNNs vs Transformer Networks

RNNs

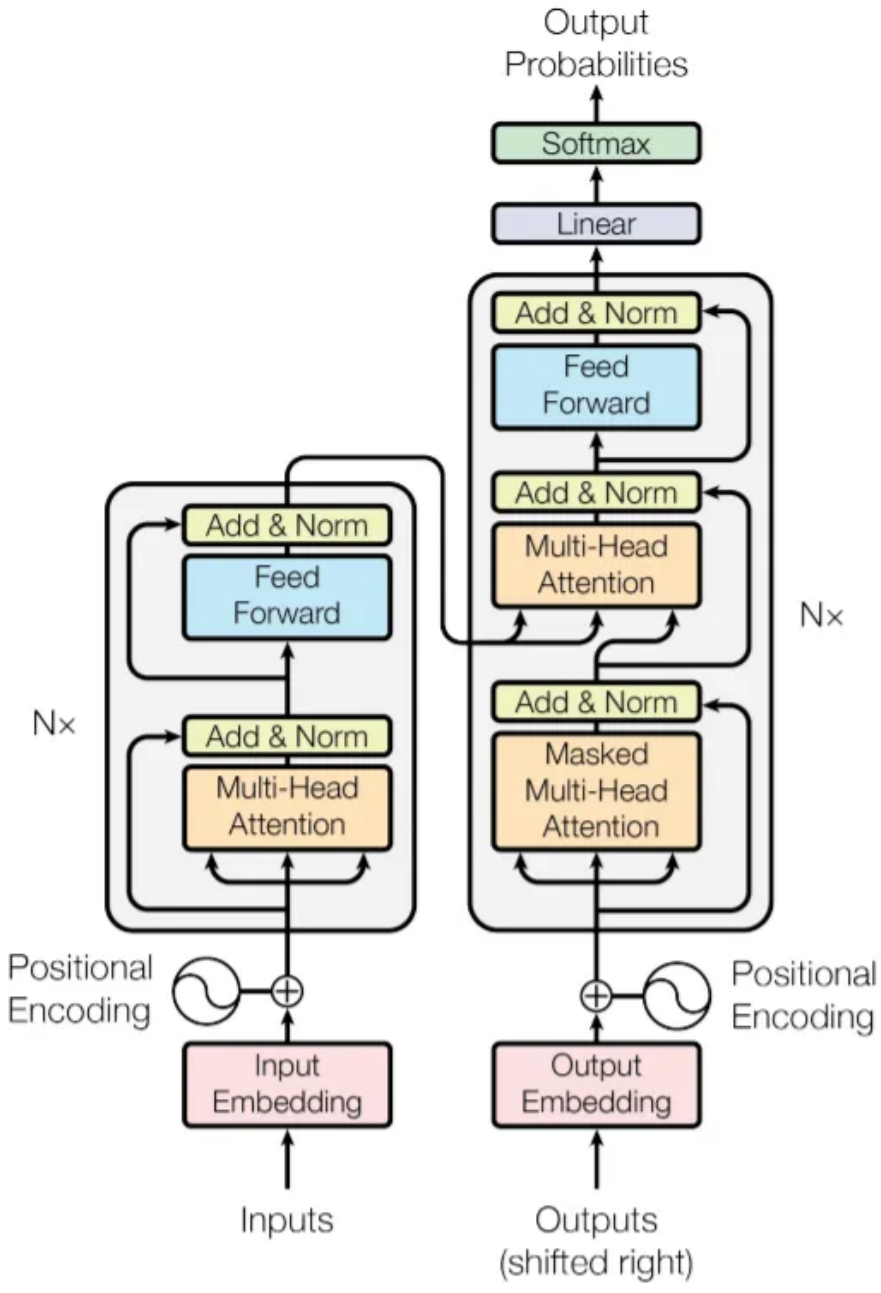
- podem não funcionar com dependências longas
- recorrência dificulta computação paralela
- podem sofrer com explosão ou desaparecimendo de gradiente

Transformer Networks

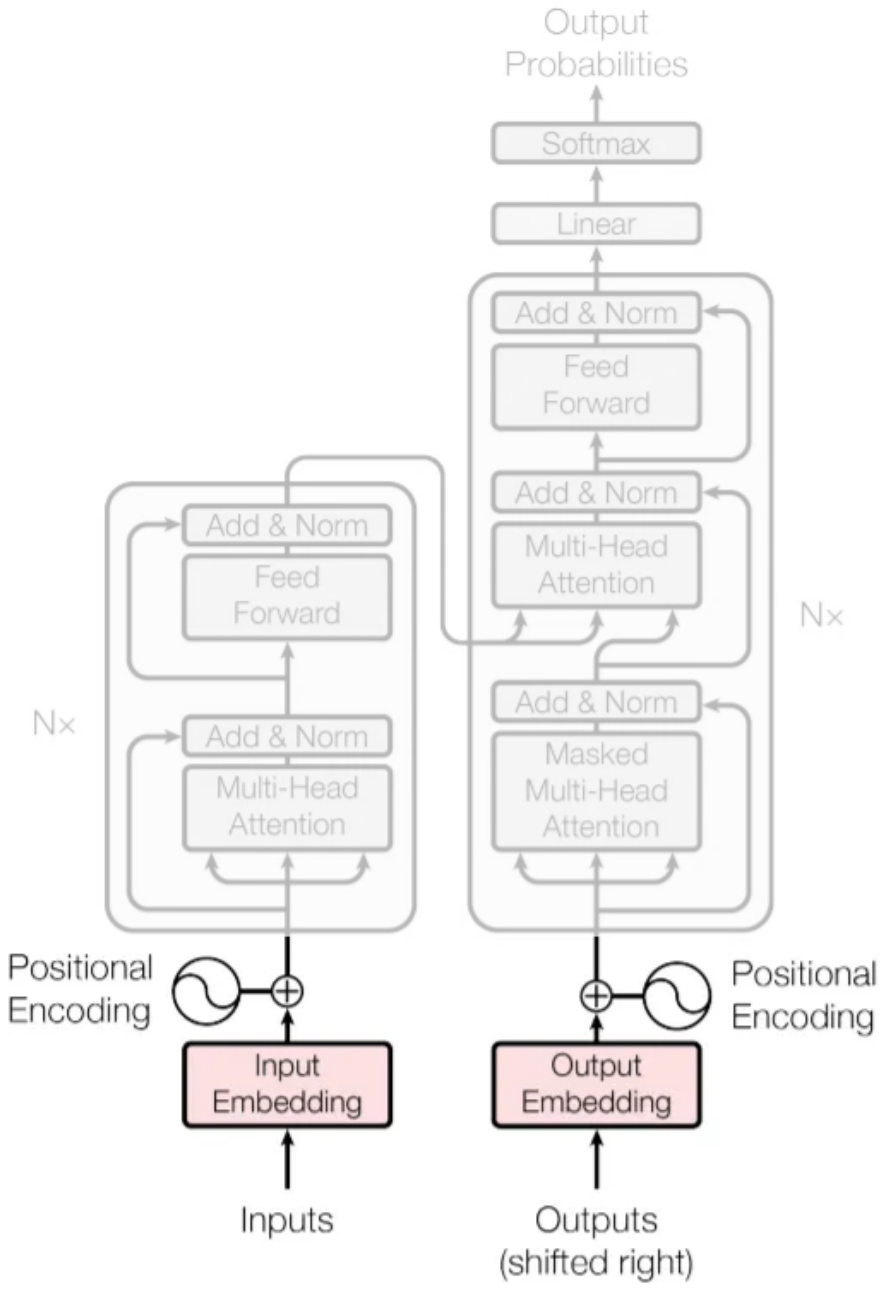
- não possui recorrência, apenas atenção
- facilita capturar dependências longas
- facilita processamento paralelo

Artigo: "Attention is all you need" Vaswani, NeurIPS 2017.

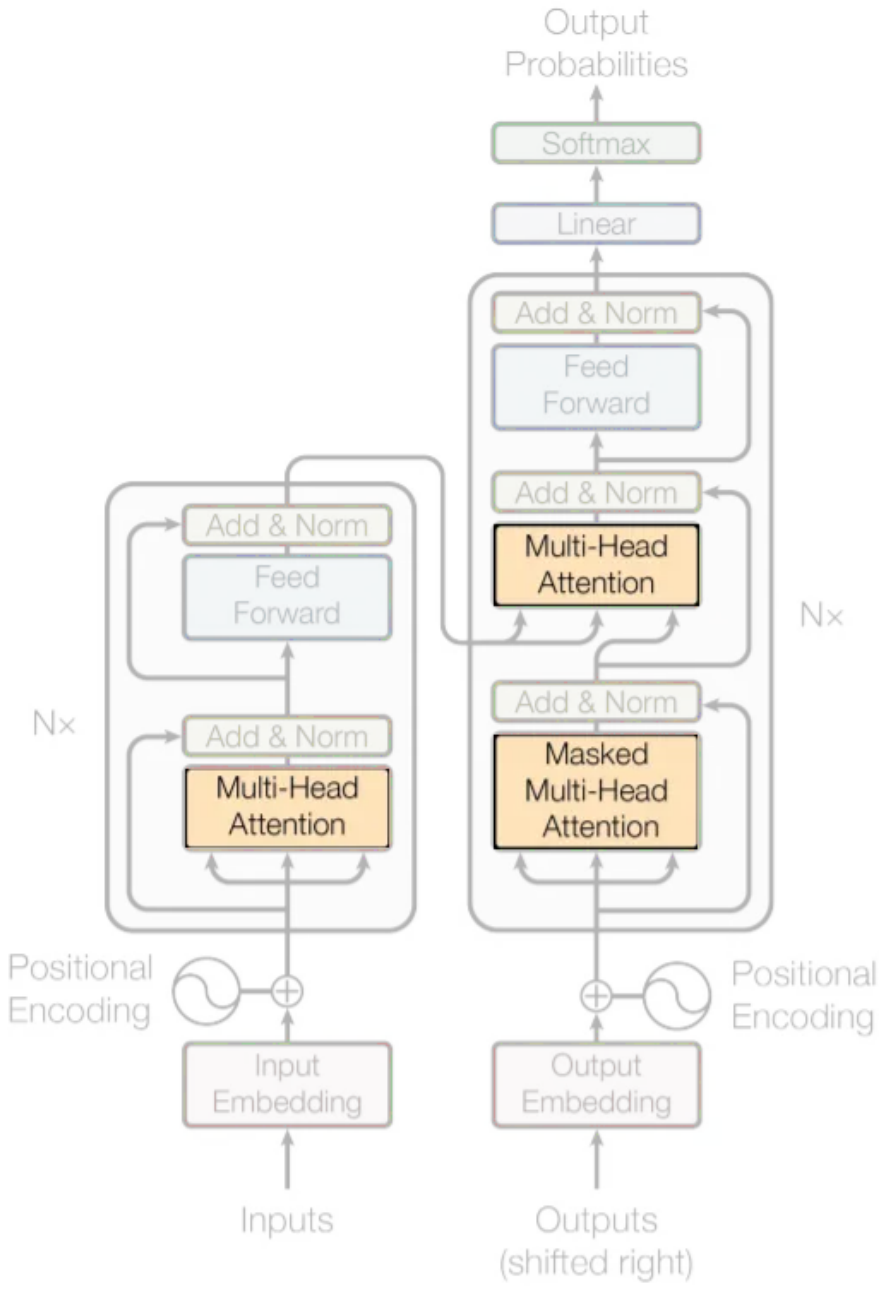
Arquitetura Transformer



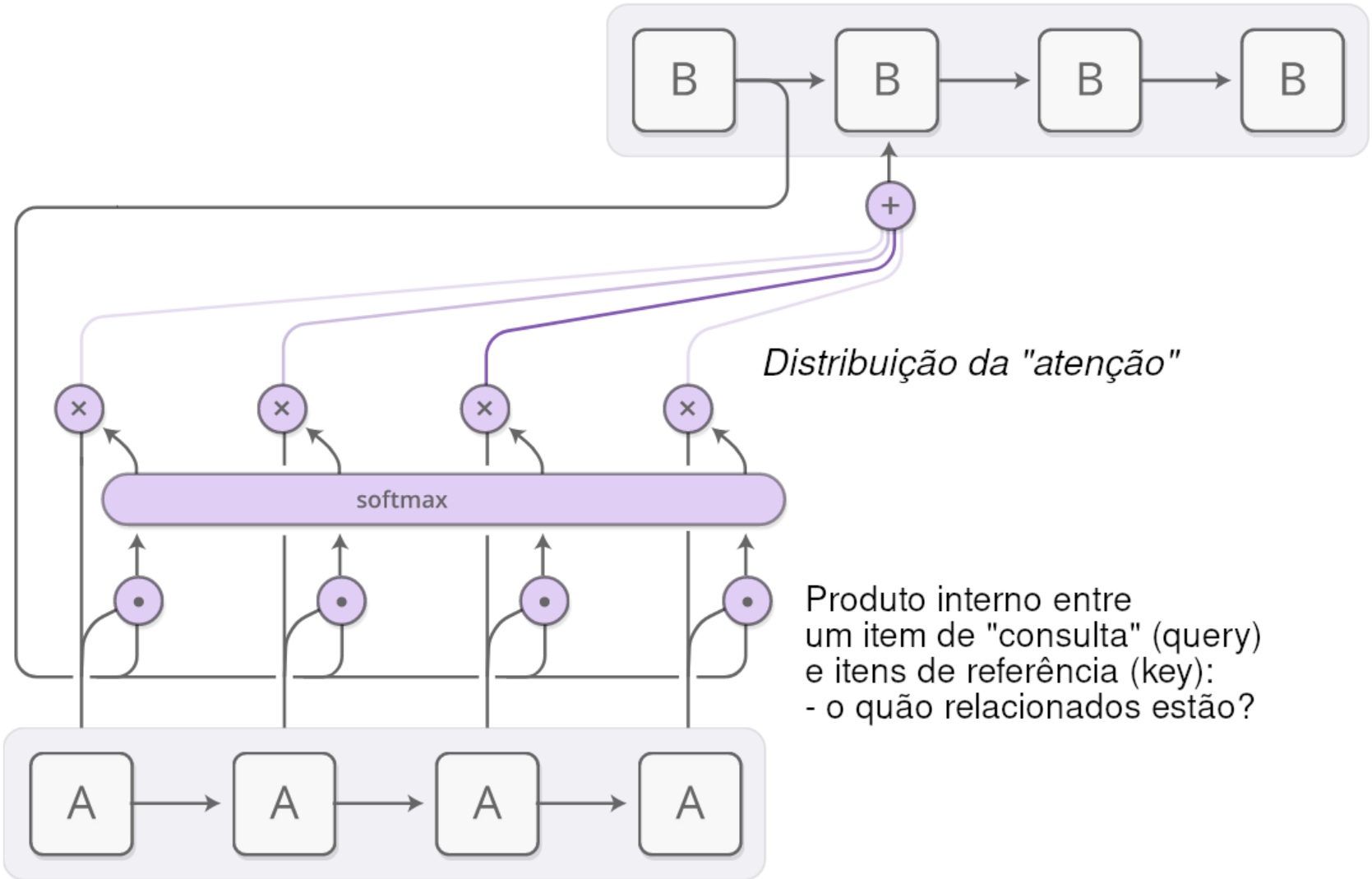
Positional Encoding: adiciona informação de posicionamento



Arquitetura Transformer: multi-atenção



Distribuição da atenção com relação ao estado atual



Mecanismo de atenção em Transformer Networks

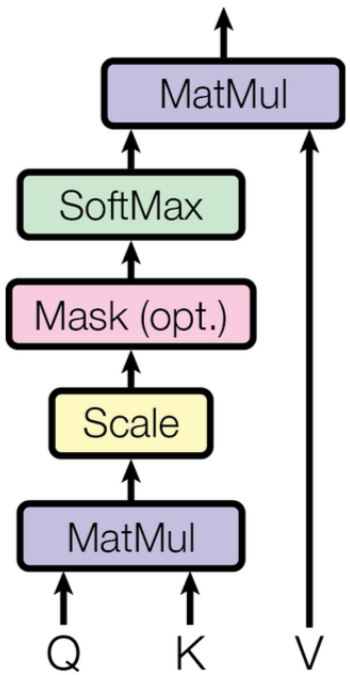
- Recuperar um valor v_i para uma consulta/query q baseada numa chave/key k_i

$$Attention(q, \mathrm{k}, \mathrm{v}) = \sum_i similarity(q, k_i) \times v_i$$

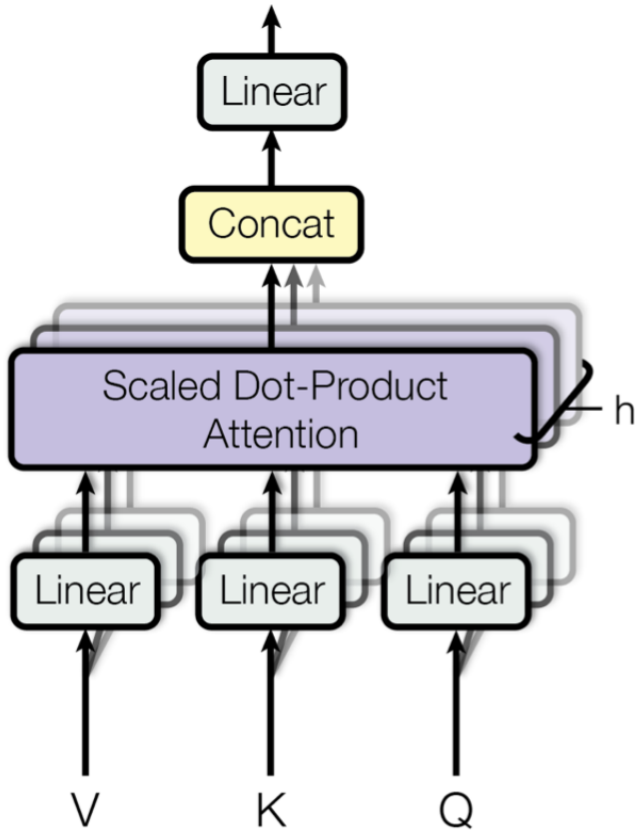
- A **similaridade** entre uma consulta e todas as chaves, ponderadas pelos valores
- Somar ao longo de todas as chaves/valores, produz uma **distribuição** de pesos relacionando consulta e todos os valores

Multi-head attention

Scaled Dot-Product Attention



Multi-Head Attention



Representações para cada palavra

1. Word embeddings: w
2. Word embeddings + Positional encoding: $\mathrm{w} + \mathrm{p} = \mathrm{e}$
3. Query: $\mathrm{Q}_h = W_Q \cdot \mathrm{e}$
4. Key: $\mathrm{K}_h = W_K \cdot \mathrm{e}$
5. Value: $\mathrm{V}_h = W_V \cdot \mathrm{e}$

h indexa as H "cabeças de atenção"

Attention e Masked Attention

Attention

$$attention(Q, K, V) = softmax\left(\frac{Q^T K}{\sqrt{d_k}}\right) V$$

d_k dimensões da chave

Masked Attention

- o decoder deve anular atenção com relação a palavras de entrada futuras, senão não há aprendizado

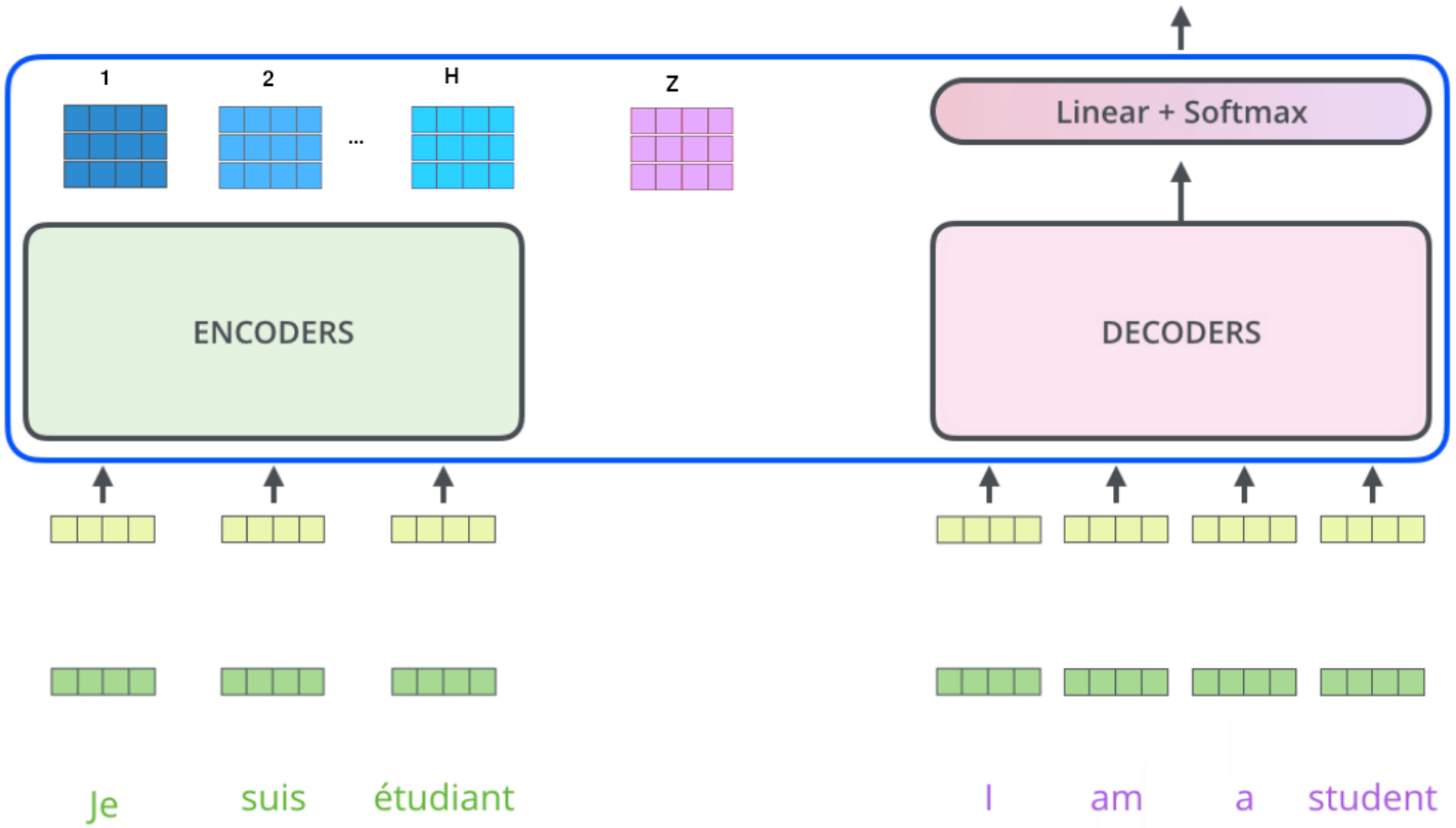
$$maskedattention(Q, K, V) = softmax\left(\frac{Q^T K + M}{\sqrt{d_k}}\right) V$$

M é uma matriz com `-inf` nas posições de palavras futuras

Representações para cada palavra

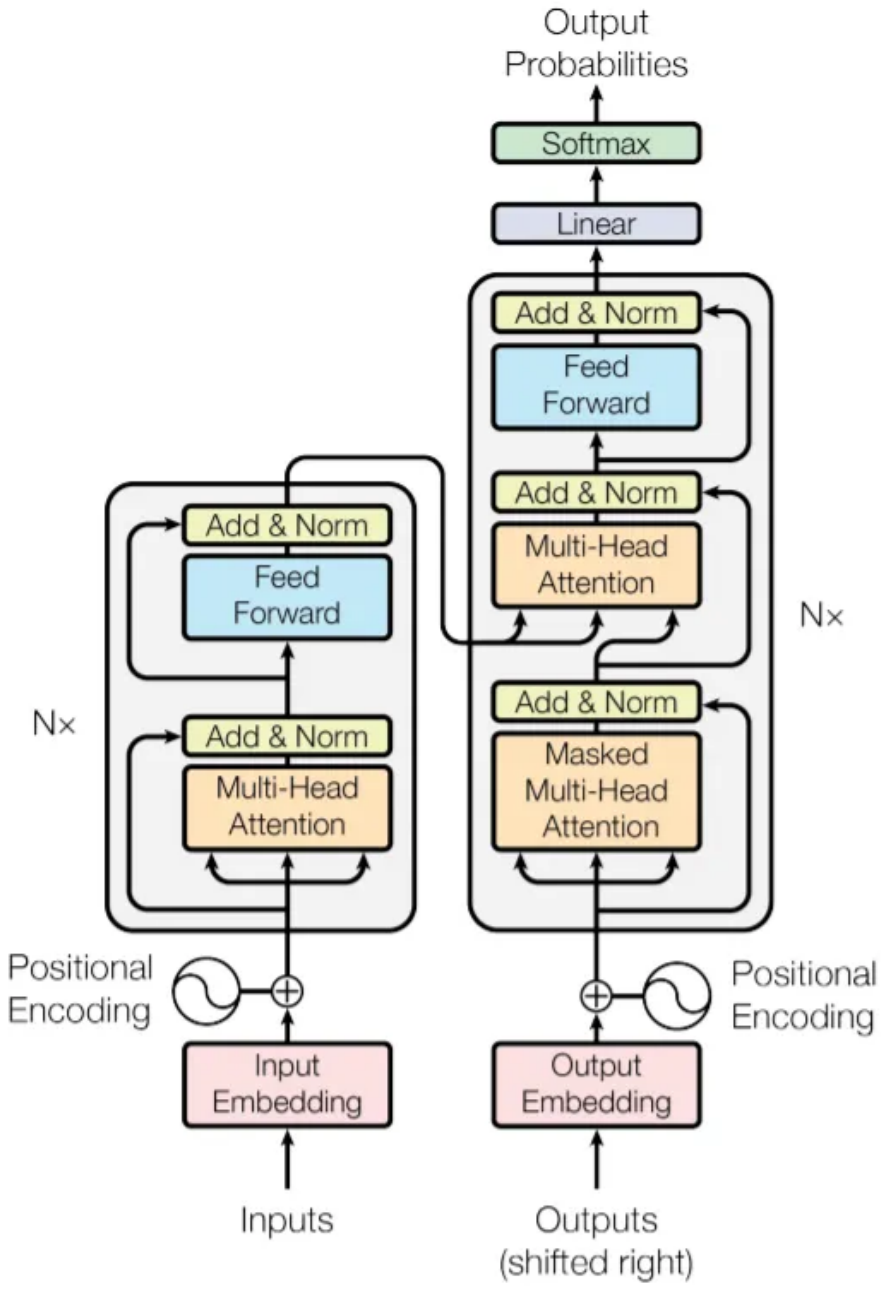
1. Word embeddings: w
2. com positional encoding: $\mathrm{w} + \mathrm{p} = \mathrm{e}$
3. Query: $\mathrm{Q}_h = W_Q \cdot \mathrm{e}_h$
4. Key: $\mathrm{K}_h = W_K \cdot \mathrm{e}_h$
5. Value: $\mathrm{V}_h = W_V \cdot \mathrm{e}_h$
6. Attention heads: $Z_h = softmax\left(\frac{Q_h^T K_h + M}{\sqrt{d_k}}\right) V_h$
7. Attention output: $Z = W_O concat((Z_1, Z_2, \cdots Z_H))$

Recorrência via "Teacher forcing" no treinamento



Adaptado de Jay Alammar

Arquitetura Transformer



Arquitetura Transformer: alguns resultados

One day, I decided to bake a loaf of bread on my own oven.

`https://transformer.huggingface.co/`

Arquitetura Transformer: alguns resultados

One day, I decided to bake a loaf of bread on my own oven. It was so simple and simple and easy.

`https://transformer.huggingface.co/`

Arquitetura Transformer: alguns resultados

One day, I decided to bake a loaf of bread on my own oven. It was so simple and simple and easy. This bread is my first time baking it in my oven.

`https://transformer.huggingface.co/`

Arquitetura Transformer: alguns resultados

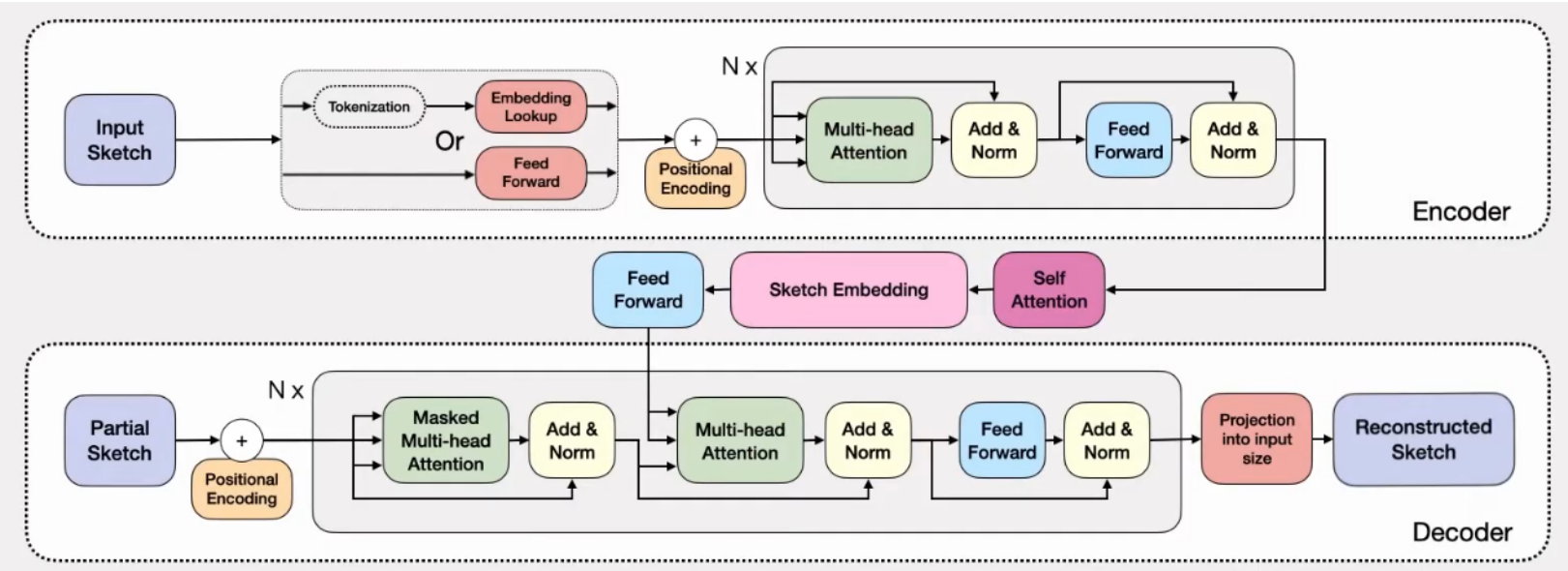
One day, I decided to bake a loaf of bread on my own oven. It was so simple and simple and easy. This bread is my first time baking it in my oven. I don't know what you will have to do to make this loaf.

`https://transformer.huggingface.co/`

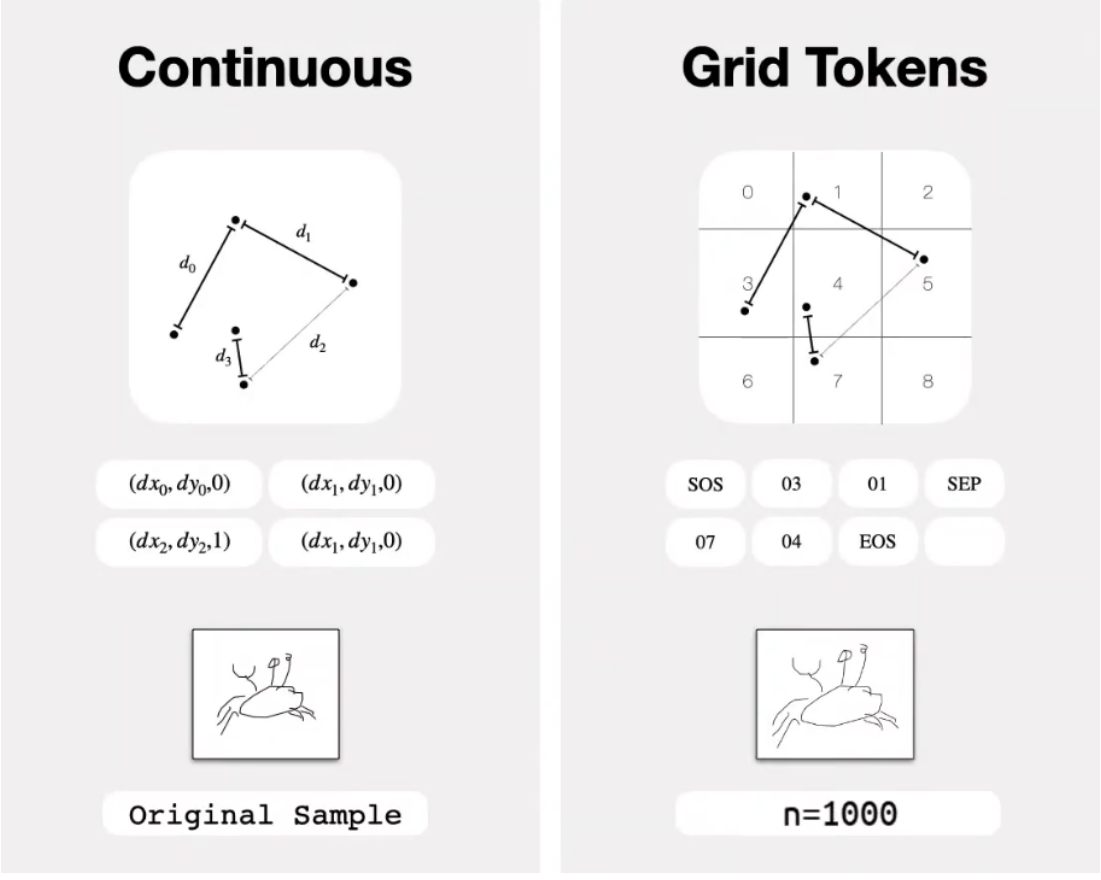
Após Transformer

- GPT e GPT2
- BERT
- Lambda Networks
- ...

Outras aplicações: para dados visuais



Outras aplicações: para dados visuais



References

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention is all you need, NeurIPS 2017
- Dzmitry Bahdanau, KyungHyun Cho and Yoshua Bengio. Neural Machine Translation By Jointly Learning To Align And Translate. ICLR 2015.
- A. Karpathy. Understanding LSTM Networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`
- C. Olah. Understanding LSTM Networks `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`