



INSIGHT

Data Science Laboratory
Federal University of Ceará

REDES NEURAIS RECORRENTES

OLÁ!

Sou Lívia Almada,

Professora da UFC no Campus de Quixadá,
doutoranda em Ciência da Computação e
Insighter..

Você pode me encontrar em
livia.almada@ufc.br

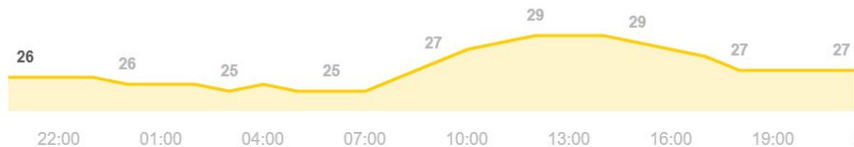
DADOS SEQUENCIAIS (SEQUÊNCIAS)



- ▶ Elementos aparecem em uma determinada ordem e não são independentes uns dos outros
 - ▶ A ordem é importante!!!

 26 °C | °F

Chuva: 7%
Umidade: 77%
Vento: 14 km/h

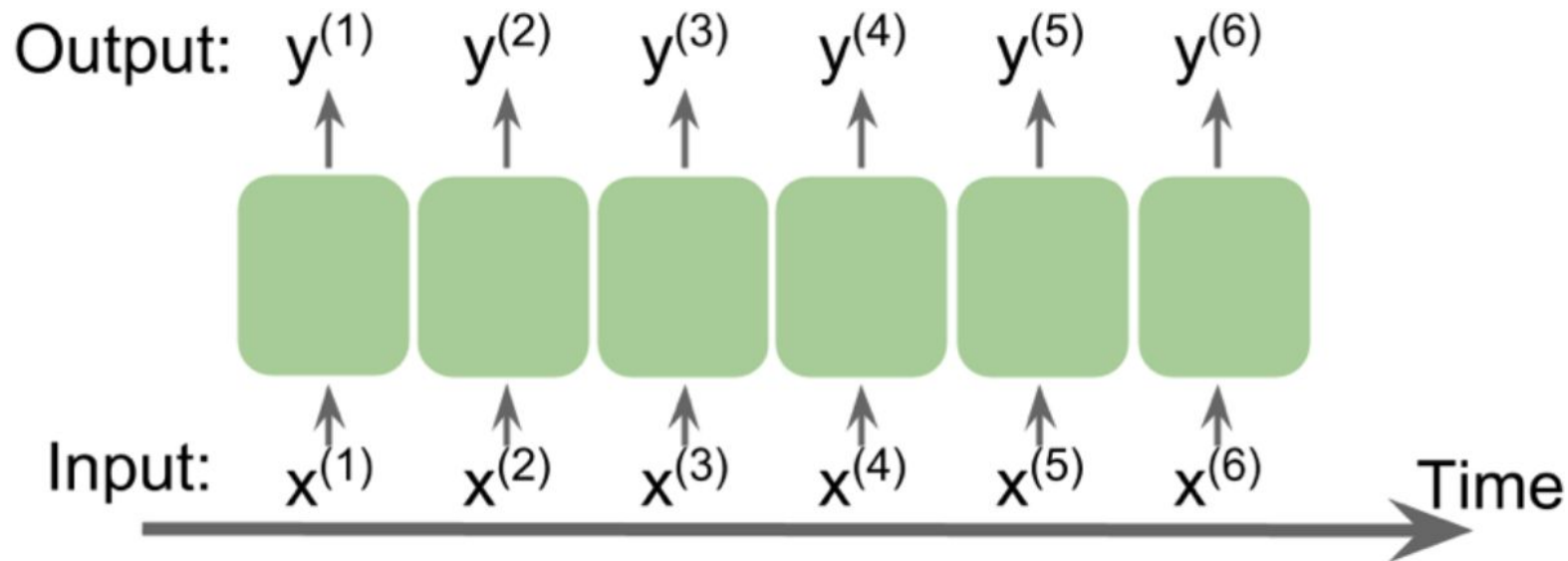
Temperatura Chuva Vento



ter	qua	qui	sex	sáb	dom	seg	ter
							
30° 24°	30° 24°	31° 23°	31° 23°	30° 23°	30° 23°	30° 24°	30° 24°

REPRESENTANDO SEQUÊNCIAS

Uma sequência de tamanho T : $x(1), x(2), x(3), \dots, x(T)$

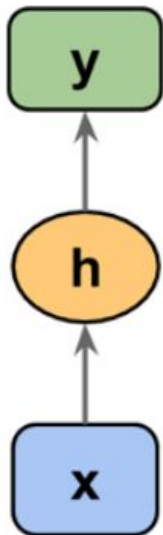


REPRESENTANDO SEQUÊNCIAS

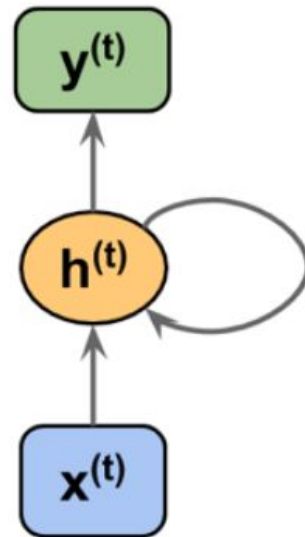
- ▶ Os modelos de rede neural padrão que vimos como MLPs e CNNs, não são capazes de lidar com ordem de amostras de entrada.
 - ▶ Não memorizam amostras vistas no passado.
- ▶ As amostras são passadas pelo feedforward e etapas de retropropagação e os pesos são atualizados independente da ordem em que a amostra é processada
- ▶ As RNNs são projetadas para sequências e são capazes de lembrar informações passadas.

ENTENDENDO A ESTRUTURA DA RNN

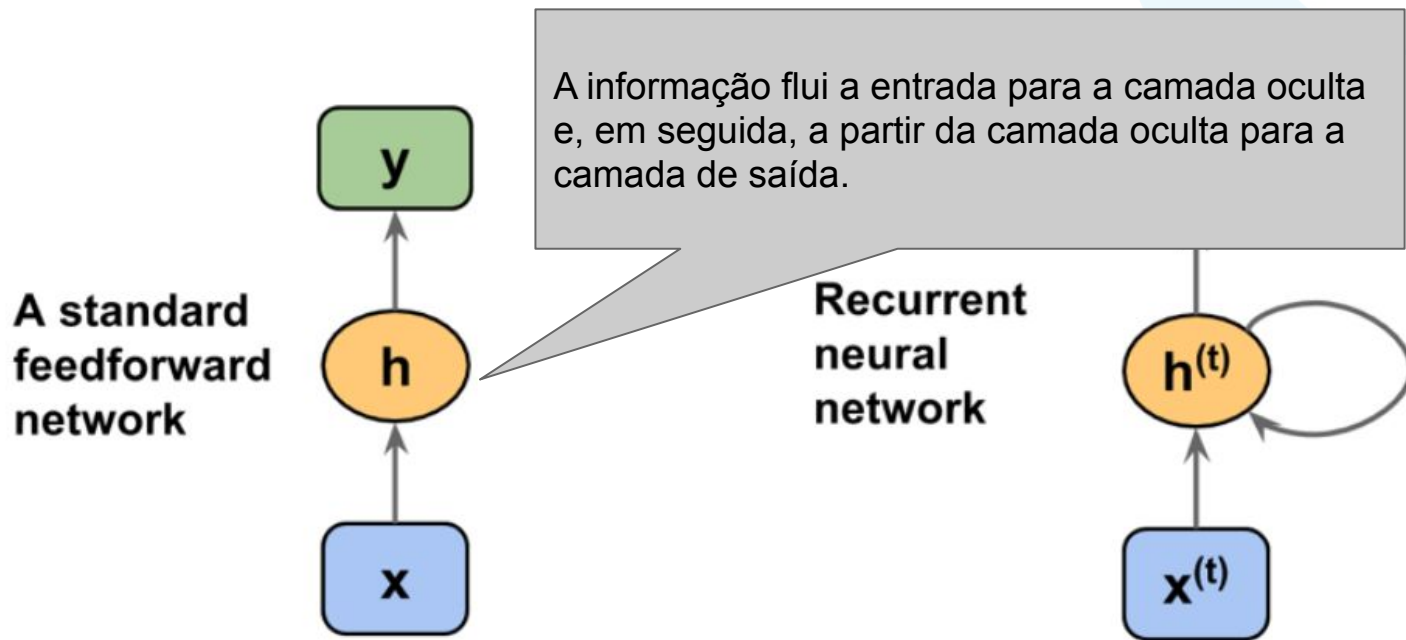
A standard
feedforward
network



Recurrent
neural
network



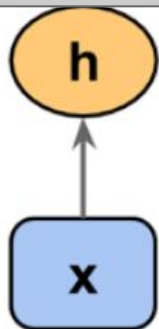
ENTENDENDO A ESTRUTURA DA RNN



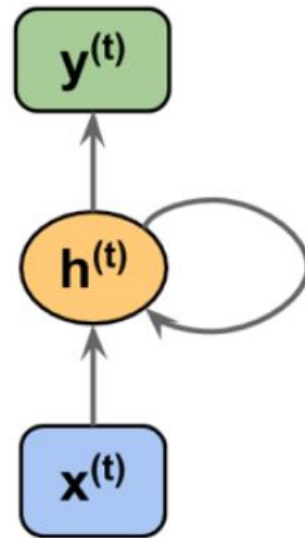
ENTENDENDO A ESTRUTURA DA RNN

A camada oculta obtém sua entrada da camada de entrada e da camada oculta do tempo anterior ($t-1$)

A standard
feedforward
network



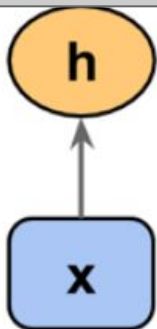
Recurrent
neural
network



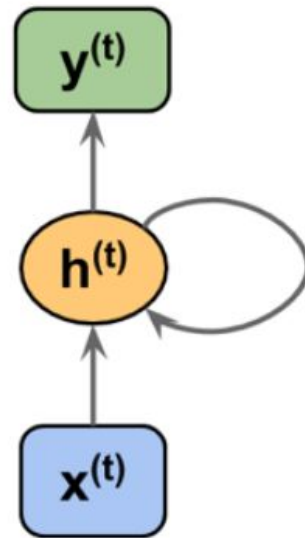
ENTENDENDO A ESTRUTURA DA RNN

O fluxo de informações em etapas de tempo adjacentes na camada oculta permite que a rede tenha uma memória de eventos passados.

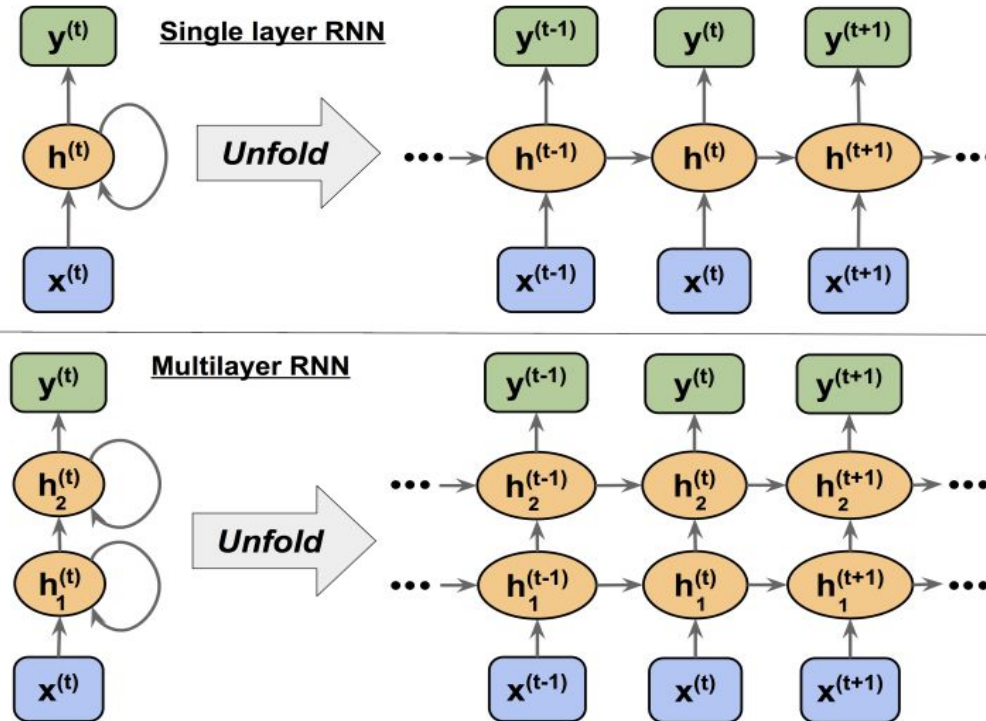
A standard
feedforward
network



Recurrent
neural
network

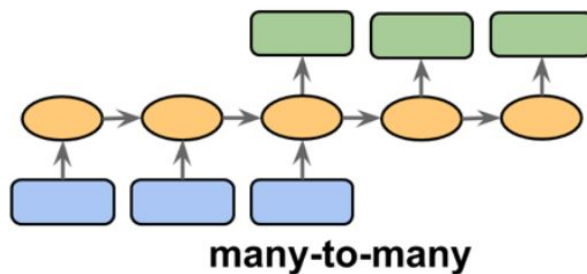
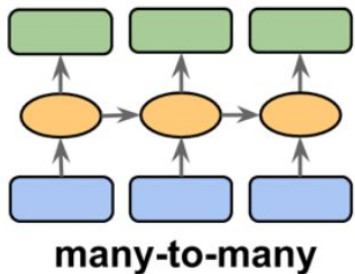


ENTENDENDO A ESTRUTURA DA RNN (uma camada e múltiplas camadas)

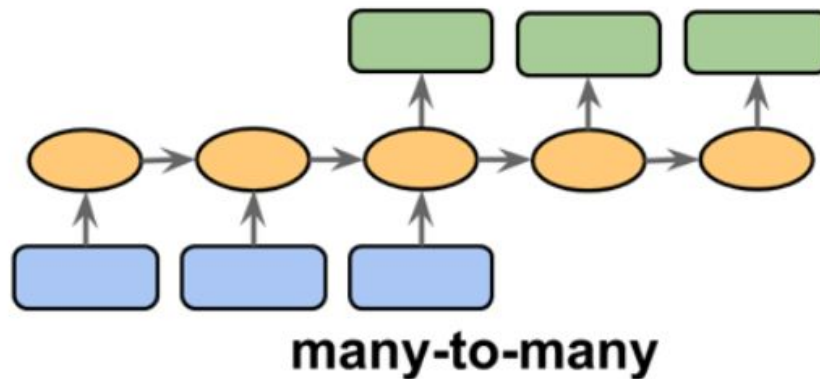
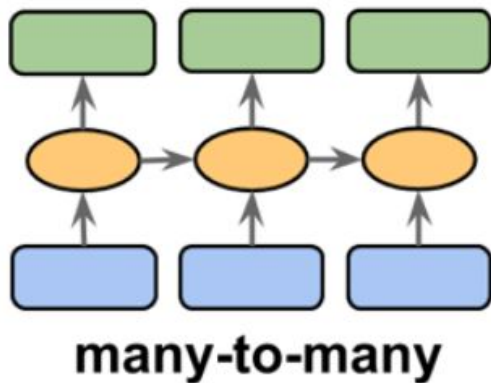


DIFERENTES CATEGORIAS DE MODELAGEM SEQUÊNCIAS

- ▶ Many-to-one: A entrada é um sequência, mas a saída não é. Ex. análise de sentimentos
- ▶ One-to-many: A entrada não é uma sequência, mas a saída é. Ex. Legenda para imagens



DIFERENTES CATEGORIAS DE MODELAGEM SEQUÊNCIAS

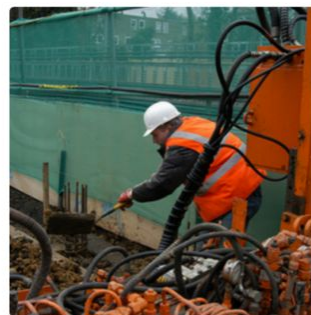


DIFERENTES CATEGORIAS DE MODELAGEM SEQUÊNCIAS

- ▶ **Many-to-one:** A entrada é um sequência, mas a saída não é. Ex. análise de sentimentos
- ▶ **One-to-many:** A entrada não é uma sequência, mas a saída é. Ex. Legenda para imagens



"man in black shirt is playing guitar."

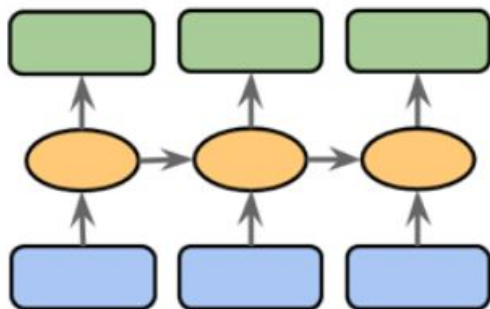


"construction worker in orange safety vest is working on road."

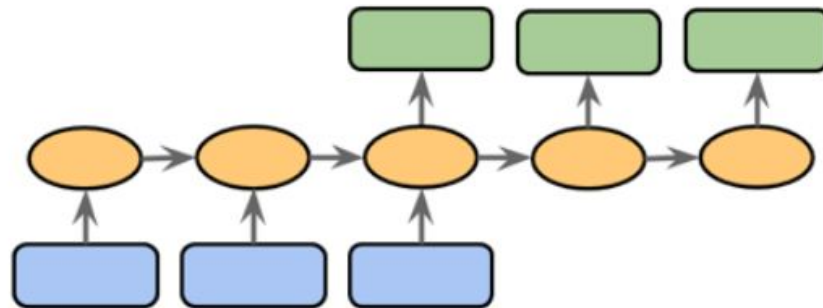


"two young girls are playing with lego toy."

DIFERENTES CATEGORIAS DE MODELAGEM SEQUÊNCIAS



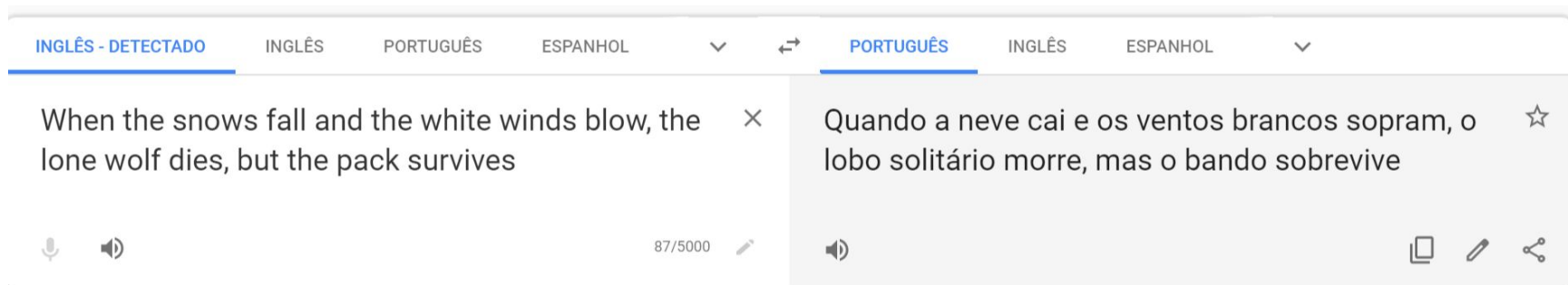
many-to-many



many-to-many

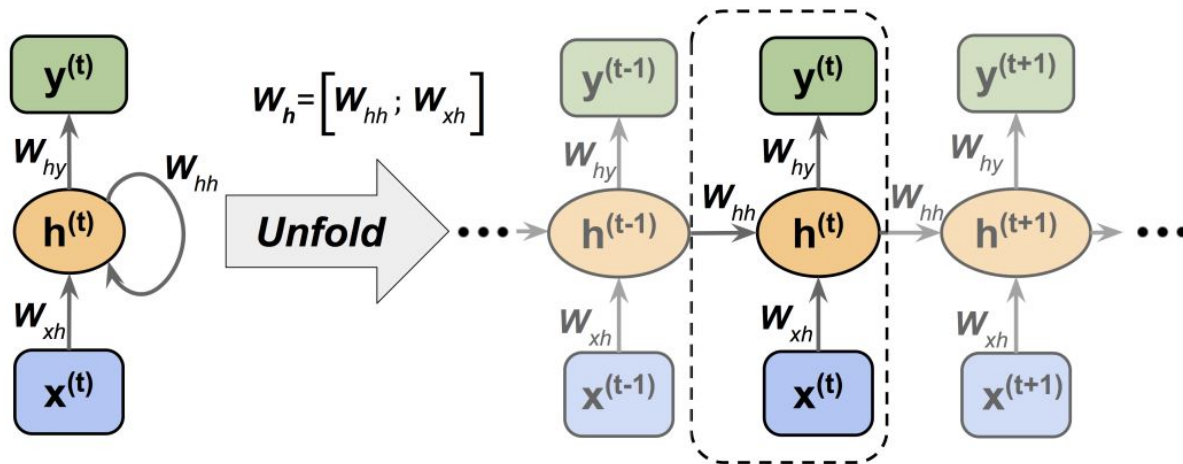
DIFERENTES CATEGORIAS DE MODELAGEM SEQUÊNCIAS

- ▶ **Many-to-many:** (1) sincronizado. Ex. classificação de vídeos, onde cada frame tem um label.
- ▶ **Many-to-many:** (2) delayed. Ex. Tradução de textos.



COMPUTANDO ATIVAÇÃO DA RNN

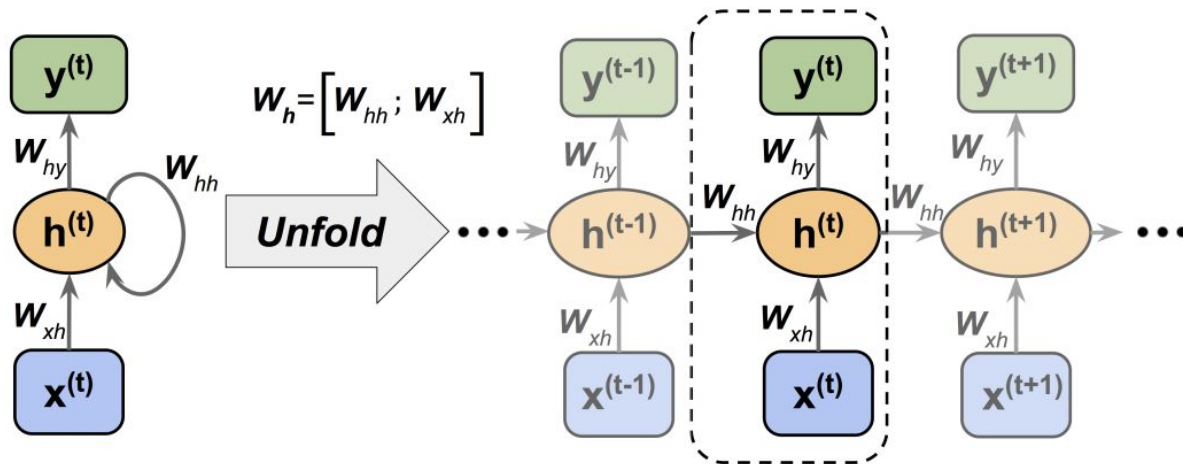
- ▶ Cada aresta direcionada (as conexões entre as caixas) na representação da RNN está associada a uma matriz de peso.
- ▶ Esses pesos não dependem do tempo t ; portanto, eles são compartilhados no eixo do tempo.



COMPUTANDO ATIVAÇÃO DA RNN

As diferentes matrizes de peso em um camada única RNN são os seguintes:

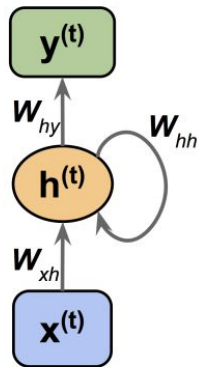
- ▶ W_{xh} : A matriz de peso entre a entrada $x(t)$ e a camada oculta h
- ▶ W_{hh} : A matriz de peso associada à aresta recorrente
- ▶ W_{hy} : A matriz de peso entre a camada oculta e a saída



COMPUTANDO ATIVAÇÃO DA RNN

- ▶ A computação das ativações é semelhante aos perceptrons multicamadas e outros tipos de neural feedforward redes.
- ▶ Para a camada oculta, a entrada z_h (pré-ativação) é computada através de uma combinação linear. Ou seja, nós calculamos a soma das multiplicações das matrizes de peso com as vetores correspondentes e adicione a unidade de polarização.

$$z_h^{(t)} = W_{xh}x^{(t)} + W_{hh}h^{(t-1)} + b_h$$

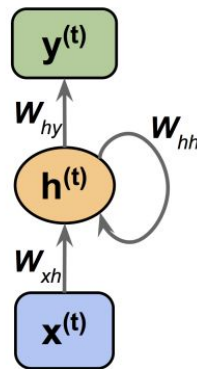


COMPUTANDO ATIVAÇÃO DA RNN

- ▶ A ativação das unidades ocultas na etapa **t** são calculadas como:

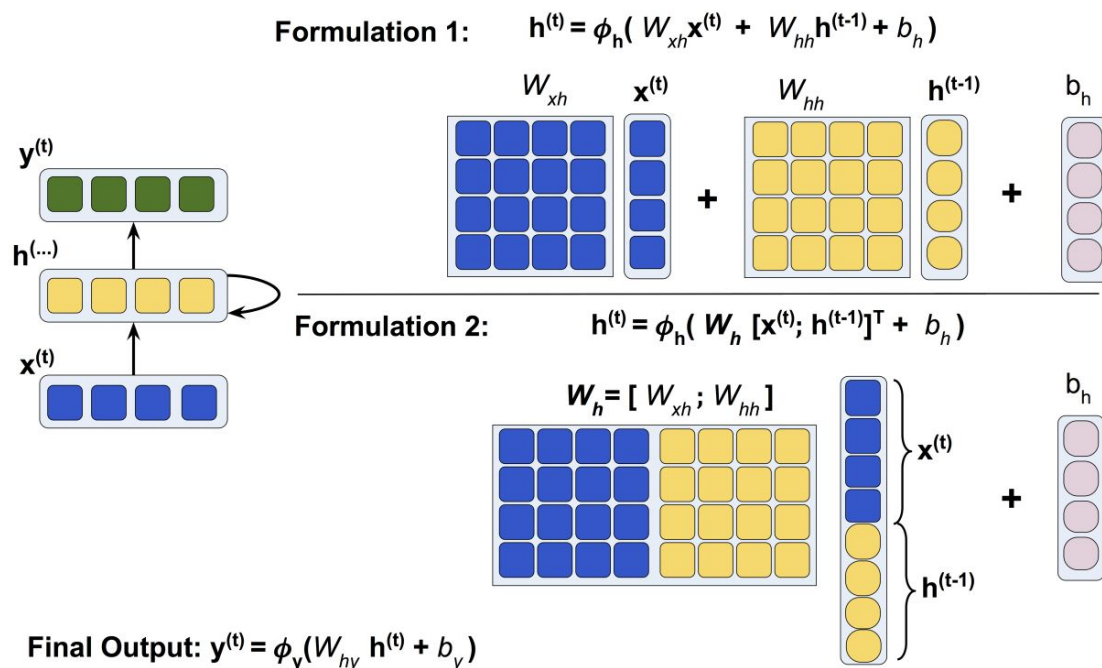
$$z_h^{(t)} = W_{xh}x^{(t)} + W_{hh}h^{(t-1)} + b_h$$

$$h^{(t)} = \phi(z_h^{(t)}) = \phi(W_{xh}x^{(t)} + W_{hh}h^{(t-1)} + b_h)$$



COMPUTANDO ATIVAÇÃO DA RNN

- Esquema do processo de computação das ativações



RNN e dados textuais

- ▶ Redes neurais trabalham apenas com dados numéricos!
- ▶ **Vetorizar o texto: processo de transformar o texto em vetores.**
 - ▶ Segmentar o texto em palavras e transformar cada palavra em um vetor
 - ▶ Segmentar o texto em caracteres e transformar cada caractere em um vetor
 - ▶ Extrair n-grams das palavras ou caracteres e transformar os n-grams em vetores

N-grams

- ▶ N-grams são grupos de até N palavras múltiplas e consecutivas, com sobreposição.
- ▶ Pode ser aplicado para caracteres ou palavras.
- ▶ Exemplo:
 - ▶ Frase: "The cat sat on the mat".
 - ▶ Decomposição em conjuntos de 2-grams: "The", "The cat", "cat", "cat sat", "sat", "sat on", "on", "on the", "the", "the mat", "mat"
- ▶ Observe que essa representação *bag-of-words* não preserva a ordem dos *tokens* (os tokens são compreendidos como conjuntos e não como sequência).

Transformando tokens em vetores

- ▶ One-hot encoding:

Vocabulary:
Man, woman, boy,
girl, prince,
princess, queen,
king, monarch



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

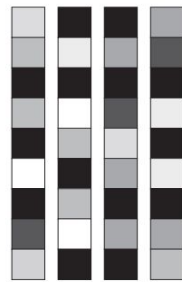
Each word gets
a 1x9 vector
representation

Transformando tokens em vetores

- ▷ Word embeddings:
 - ▶ Vetores inicialmente são obtidos com representação one-hot (binários e com alta dimensão)
 - ▶ Novos vetores são obtidos a partir dessa representação
 - ◆ Word2Vec (Miklov et. al, 2013)
<https://arxiv.org/abs/1301.3781>
 - ◆ Embedding Layers (Keras)
<https://keras.io/layers/embeddings/>

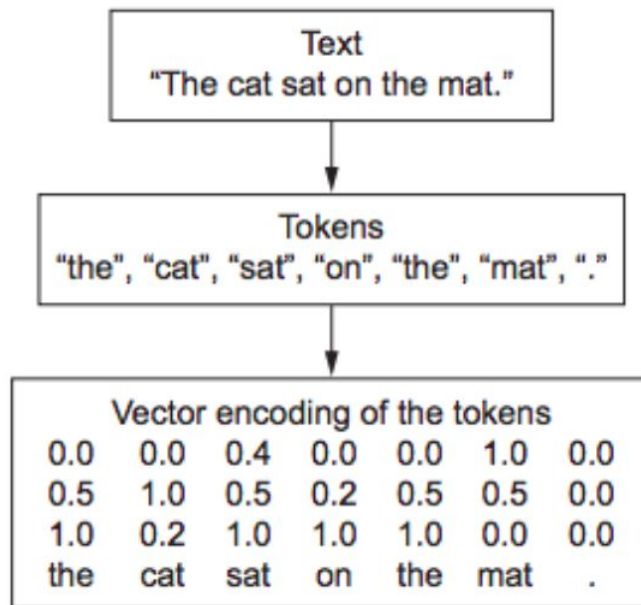


One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Processamento dos textos



N-grams

- ▶ Redes neurais trabalham apenas com dados numéricos!
- ▶ **Vetorizar o texto: processo de transformar o texto em vetores.**
 - ▶ Segmentar o texto em palavras e transformar cada palavra em um vetor
 - ▶ Segmentar o texto em caracteres e transformar cada caractere em um vetor
 - ▶ Extrair n-grams das palavras ou caracteres e transformar cada n-gram em um vetor
 - ◆ n-grams são grupos de n palavras múltiplas e consecutivas, com sobreposição

Mão na massa!

Análise de sentimentos com RNN



LSTM

- ▶ Problema da versão vanilla das RNNs: teoricamente capazes de retrainar a informação sobre os inputs vistos em muitas etapas anteriores no tempo t . Entretanto, na prática, dependências de longas sequências são impossíveis de treinar.
- ▶ O motivo é o problema de *vanishing gradient*: em alguns casos, o gradiente vai desaparecendo, impedindo o peso de mudar seu valor. Na pior das hipóteses, isso pode impedir completamente a rede neural de treinamento adicional.
- ▶ Esse efeito também é observado em redes feedforward com muitas camadas.

Vanishing gradient

- ▶ Backpropagation through time, ou BPTT, introduz alguns novos desafios. A derivação dos gradientes pode ser um pouco complicada, mas a ideia básica é que a perda global L é a soma de todas as funções de perda em tempos $t = 1$ para $t = T$:

$$L = \sum_{t=1}^T L^{(t)}$$

Como a perda no tempo $1:t$ depende das unidades ocultas em todas as etapas anteriores $1:t$, o gradiente será calculado da seguinte forma:

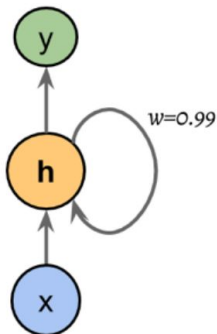
$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial \mathbf{y}^{(t)}} \times \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \times \left(\sum_{k=1}^t \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \times \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

Vanishing gradient

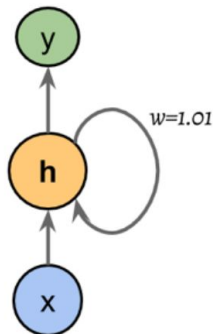
$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial \mathbf{y}^{(t)}} \times \frac{\partial \mathbf{y}^{(t)}}{\partial \mathbf{h}^{(t)}} \times \left(\sum_{k=1}^t \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \times \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

Fator multiplicativo é um produto de $t-k$ iterações.
 $t-k$ iterações de dependências de cadeias longas.

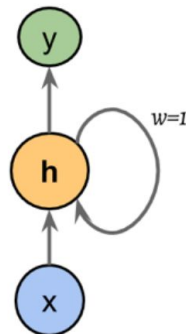
Vanishing gradient: $|w_{hh}| < 1$



Exploding gradient: $|w_{hh}| > 1$



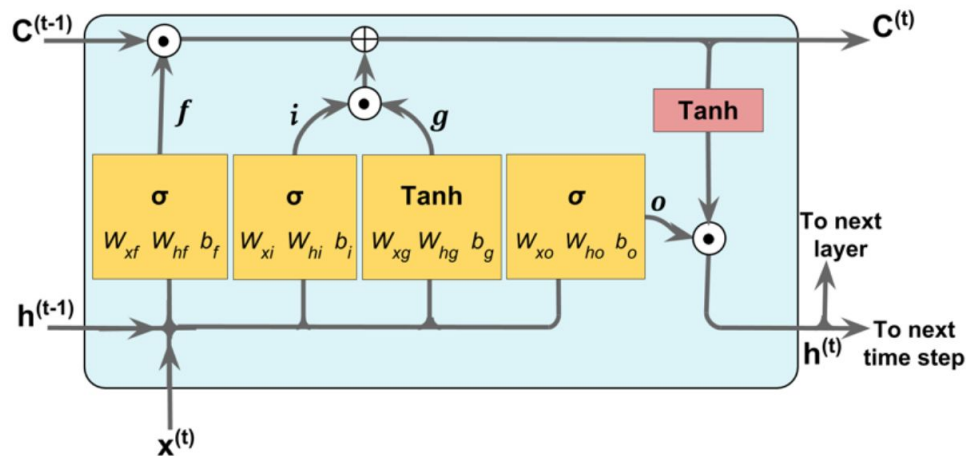
Desirable: $|w_{hh}| = 1$



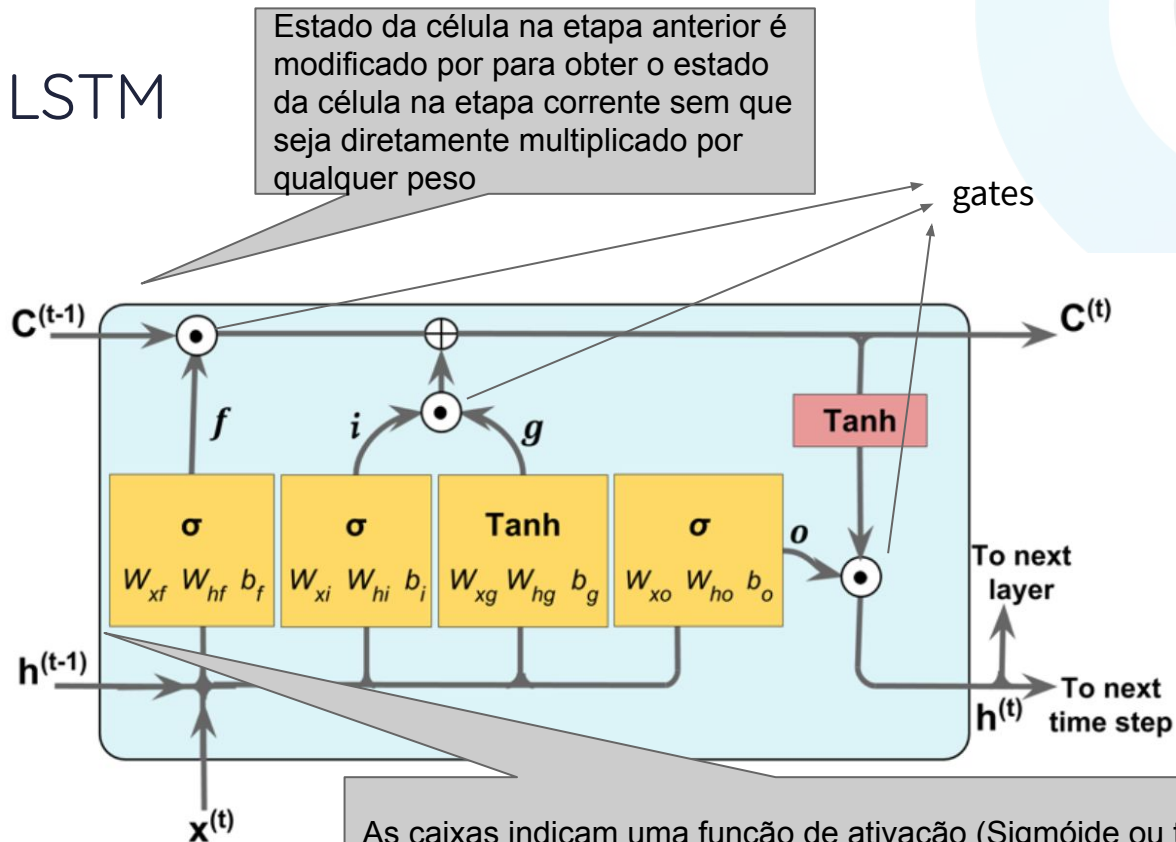
Vanishing gradient: Soluções

- ▶ Truncated backpropagation through time (TBPTT):
 - ▶ **limita o número de etapas** do backpropagation
- ▶ Long short-term memory (LSTM):
 - ▶ Aplicada com sucesso na modelagem de longas sequências e lidar com sucesso com o problema do ***vanish gradient***.

- ▶ Camadas ocultas são células de memória
- ▶ Em cada célula de memória, existe uma aresta recorrente que tem um peso desejável (**$w=1$**)
- ▶ Os valores associados com a aresta recorrente são chamados estado da célula

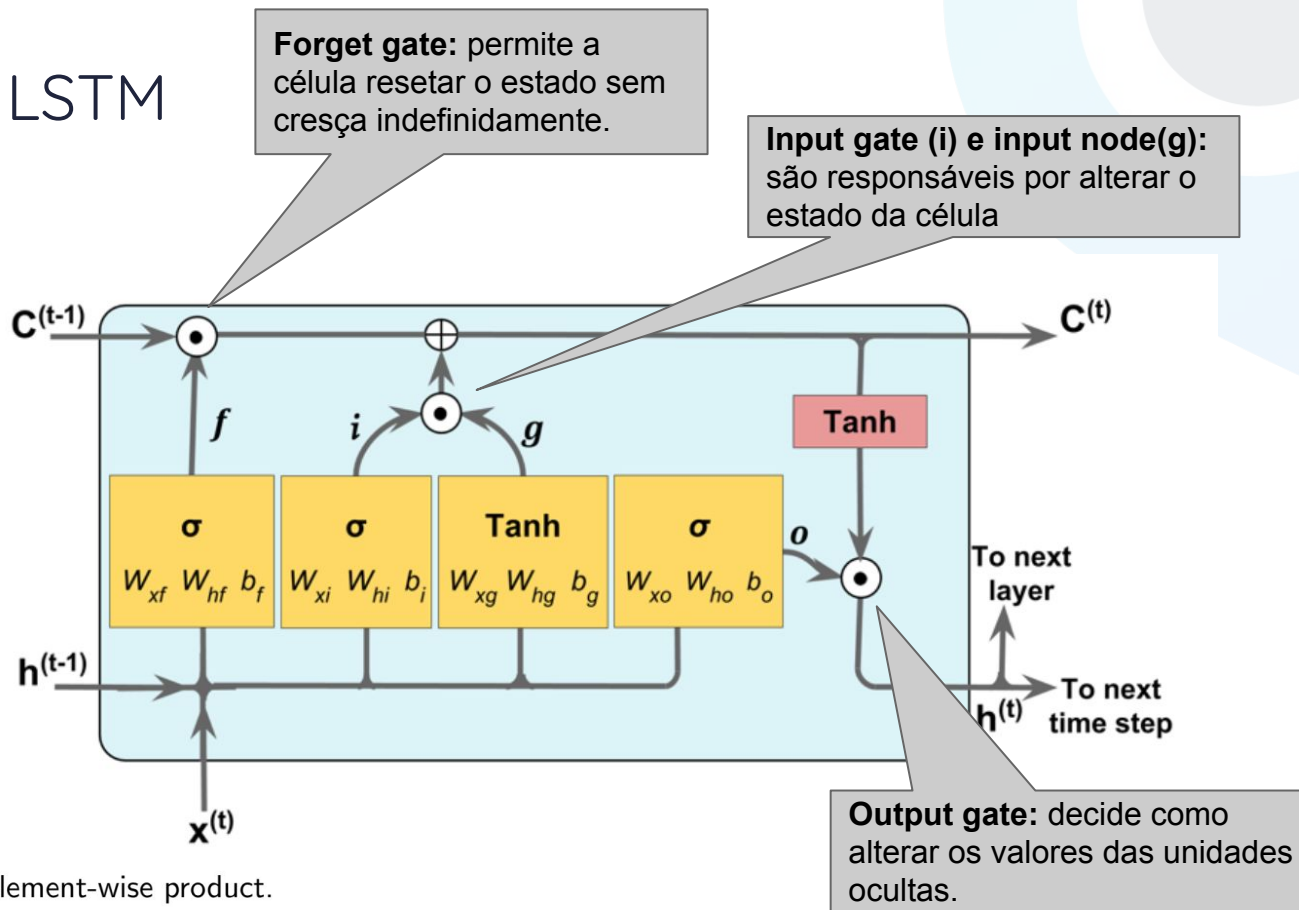


Células LSTM



As caixas indicam uma função de ativação (Sigmóide ou tangente hiperbólica) e o conjunto de pesos. Elas aplicam uma combinação linear através da multiplicação de matriz-vetor sobre as entradas.

Células LSTM

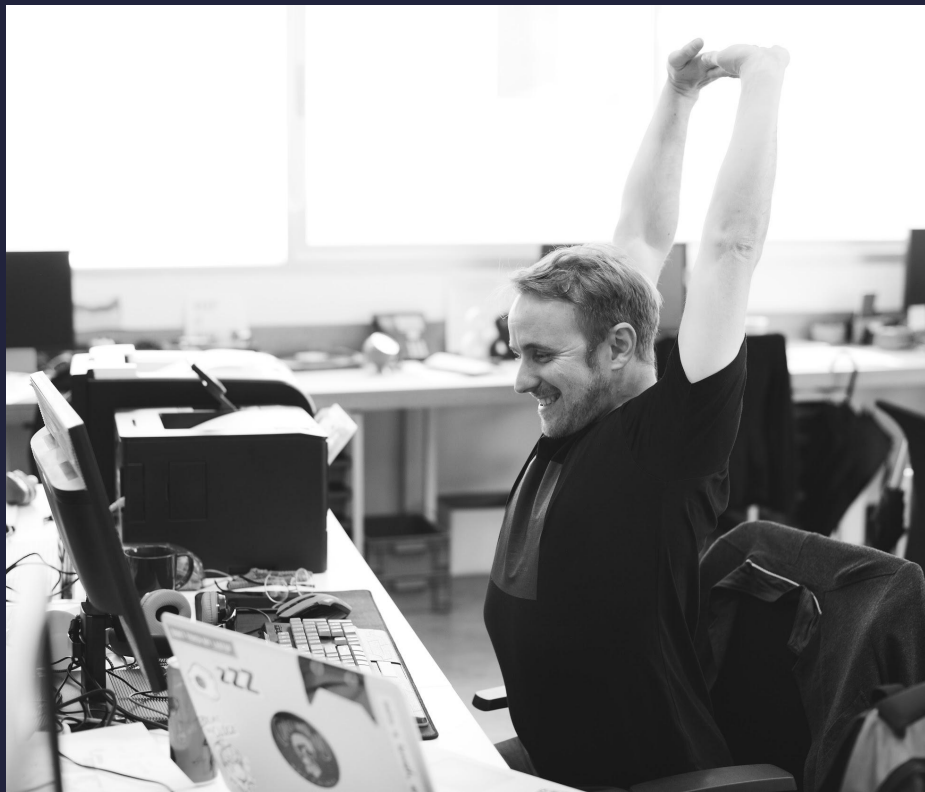


(\odot) refers to the element-wise product.

(\oplus) refers to the element-wise addition/summation.

LSTM

- ▶ A estrutura da LSTM é relativamente complexa
- ▶ A boa notícia: Keras tem tudo (tudo) implementado e podemos definir nossa LSTM facilmente!
- ▶ Apenas lembre-se do que a LSTM significa: permite que informação passada seja reinjetada a qualquer tempo, tratando do problema do *vanishing-gradient*.



OBRIGADO!

Dúvidas?