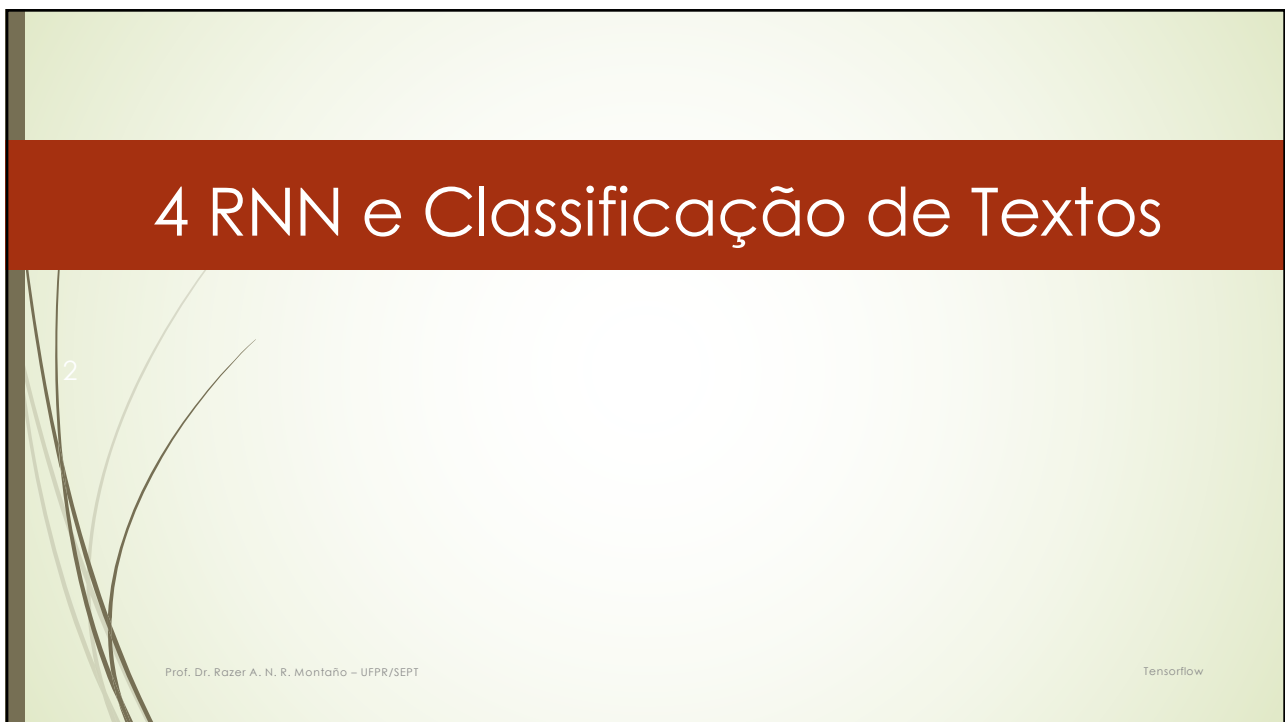




1



2

3

4.1 RNN - Redes Neurais Recorrentes

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

3

4

RNN – Redes Neurais Recorrentes

- Séries temporais
 - Preços de produtos ao longo de um ano
 - Temperatura diária ao longo do ano
 - Valores de ações por dia
 - Salário mensal de uma pessoa
 - Acidentes ocorridos por dia em uma rodovia
 - Valor do Bitcoin

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

4

5

RNN – Redes Neurais Recorrentes

- Valor do BITCOIN em 1 mês



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

5

6

RNN – Redes Neurais Recorrentes

- Tipos de previsões
 - **One-Step** : Previsão de um único valor no tempo (t)
 - Só prediz um valor no futuro
 - Pode usar uma ou todas as informações
 - **Multi-Step**: Previsão de vários valores ($t, t+1, t+2$, etc)
 - Prediz vários valores de uma vez
 - Pode ser feito:
 - Predizendo todos de uma vez
 - Predizendo t , adicionando ao modelo para predizer $t+1$, etc
 - Autoregressivo: faz-se uma previsão e a saída volta ao modelo

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

6

7

RNN – Redes Neurais Recorrentes

- Seja a seguinte sequência de valores

10 20 30 40 50 60 70 80 90 100

- Predição One-Step:** Então os dados precisam ser preparados para ter o seguinte formato

10, 20, 30 => 40

20, 30, 40 => 50

30, 40, 50 => 60

40, 50, 60 => 70

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

7

8

RNN – Redes Neurais Recorrentes

- Predição Multi-Step:** Exemplo, 3 valores no futuro

?, ?, 10 => 20, 30, 40

?, 10, 20 => 30, 40, 50

10, 20, 30 => 40, 50, 60

20, 30, 40 => 50, 60, 70

30, 40, 50 => 60, 70, 80

40, 50, 60 => 70, 80, 90

50, 60, 70 => 80, 90, 100

60, 70, 80 => 90, 100, ?

70, 80, 90 => 100, ?, ?

80, 90, 100 => ?, ?, ?

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

8

9

RNN – Redes Neurais Recorrentes

- Assim, as dimensões dos dados devem seguir $N \times T \times D$:
- D = quantidade de informação em cada leitura simples
 - Ex. 1 no caso do preço de uma ação
 - Ex. 2 no caso de GPS (latitude e longitude)
 - Etc
- T = tamanho da janela usada para previsão
 - Ex. 10 leituras para prever a próxima
 - Ex. no exemplo anterior, T era 3
- N = número de janelas disponíveis na série temporal
 - Ex. Com 100 leituras de ações, quantas janelas de tamanho 10 estão disponíveis?
 - $100 - 10 + 1 = 91$ janelas

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

9

10

RNN – Redes Neurais Recorrentes.

- Se as sequências possuem tamanho diferentes
 - Como o caso de *reviews* (avaliações), deve-se padronizar (*padding*)
 - Transformar todas as janelas para terem o mesmo tamanho
 - Ou cortar janelas grandes
 - Ou preencher com zeros (p.ex) para ter o mesmo tamanho

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

10

11

Modelos Autorregressivos: One-Step

- São modelos cujo valor no tempo t é baseado nos valores nos tempos $t-1$, $t-2$, etc

$$\hat{x}_t = w_0 + w_1x_{t-1} + w_2x_{t-2} + \dots + w_px_{t-p}$$

- **One-Step:** Quando se quer prever um valor à frente, OK

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

11

12

Modelos Autorregressivos : Multi-Step

$$\hat{x}_t = w_0 + w_1x_{t-1} + w_2x_{t-2} + \dots + w_px_{t-p}$$

- **Multi-Step:** Mais de um, NOK
 - Ex: Tem-se **dia1**, **dia2** e **dia3** – quer-se prever **dia4**, **dia5** e **dia6**
- Com o modelo treinado:
 - Usar: **dia1**, **dia2**, **dia3** para prever dia4 – OK
 - Usar: **dia2**, **dia3**, **dia4** para prever dia5 – NOK (não se conhece o dia4)
 - Usar: **dia3**, **dia4**, **dia5** para prever dia6 – NOK (não se conhece o dia 4 e dia5)

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

12

13

Modelos Autorregressivos : Multi-Step

- Pode-se usar os valores preditos como entrada

$$\widehat{x}_4 = w_0 + w_1x_1 + w_2x_2 + w_3x_3$$

$$\widehat{x}_5 = w_0 + w_1x_2 + w_2x_3 + w_3\widehat{x}_4$$

$$\widehat{x}_6 = w_0 + w_1x_3 + w_2\widehat{x}_4 + w_3\widehat{x}_5$$

- Portanto, não é possível só jogar os valores para predição
 - Deve-se fazer um laço adicionando os valores recém preditos
- Conhecido como : **Recursive Multi-step Forecast**

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

13

14

Modelos Autorregressivos : Multi-Step.

- Outras formas de predição multi-step:
 - **Direct Multi-step Forecasting**: Criar um modelo para cada passo:

$$predição_t = modelo_1(entradas)$$

$$predição_{t+1} = modelo_2(entradas)$$

$$predição_{t+2} = modelo_2(entradas)$$

- **Direct-Recursive Hybrid**: Une as duas estratégias vistas, um modelo para cada passo no tempo, usando valores previamente preditos
- **Multiple Output**: criar um modelo que prediz todos os valores de uma só vez

$$predição_t, predição_{t+1}, predição_{t+2} = modelo(entradas)$$

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

14

15

Rede Neural



$$h = f(w_h^T x + b_h)$$

$$\hat{y} = f(w_o^T h + b_o)$$

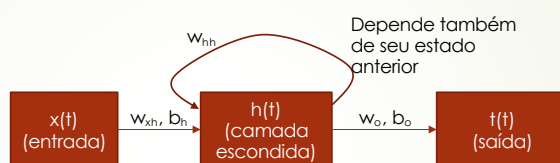
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

15

16

RNN – Rede Neural Recorrente



$$h_t = f(w_{xh}^T x_t + w_{hh}^T h_{t-1} + b_h)$$

$$\hat{y}_t = f(w_o^T h_t + b_o)$$

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

16

17

Comparação

Rede Neural

$$h = f(w_h^T x + b_h)$$

$$\hat{y} = f(w_o^T h + b_o)$$

Rede Neural Recorrente

$$h_t = f(w_{xh}^T x_t + w_{hh}^T h_{t-1} + b_h)$$

$$\hat{y}_t = f(w_o^T h_t + b_o)$$

Single Recurrent Unit

Elman Unit

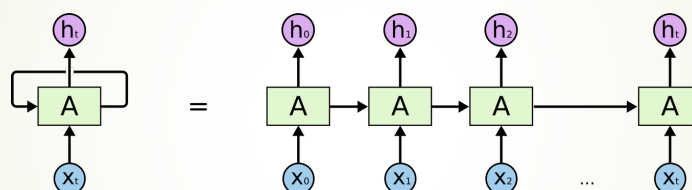
Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

17

18

RNN – Rede Neural Recorrente – Desenrolada.



Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

18

19

SimpleRNN

- **SimpleRNN** no TensorFlow/Keras
- Camada densa (totalmente conectada)
- Saída é realimentada na entrada
 - T – tamanho da sequência a ser entrada na rede
 - D – número de atributos em cada dado
 - M – número de unidades escondidas
 - K – número de nodos de saída

```
i = Input(shape=(T, D))
x = SimpleRNN(M)(i)
x = Dense(K)(x)
```

```
model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

19

20

SimpleRNN

- Problema
 - Dependências de valores
 - Ao longo do tempo se perdem valores

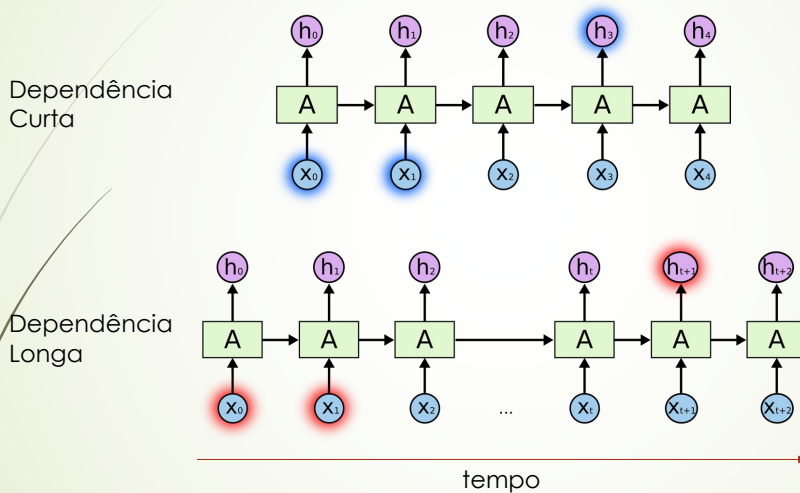
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

20

21

Dependências Curtas e Longas



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

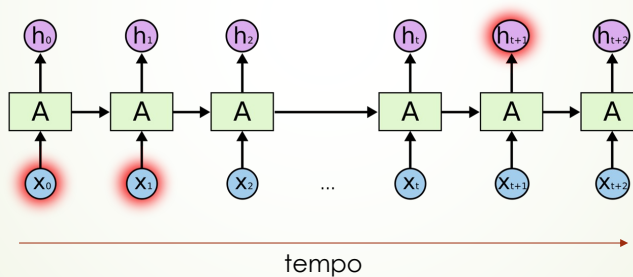
Tensorflow

21

22

Dissipação do Gradiente.

- Em dependências muito longas ao longo do tempo, muitas multiplicações são feitas
- Ex: $0,2 * 0,2 * 0,3 * 0,4 * 0,2 * 0,2 = 0,000192$
- Uma das maneiras de resolver é com outros tipos de redes: **LSTMs**



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

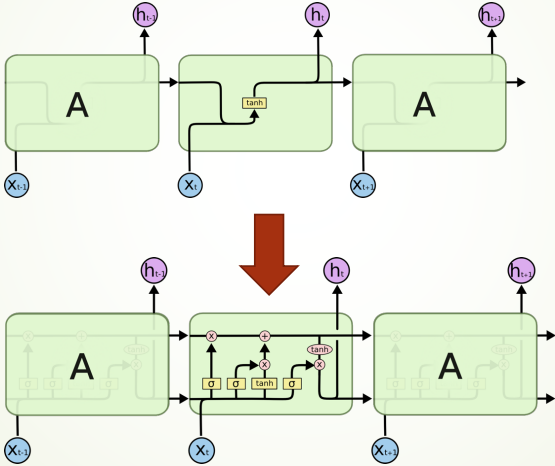
Tensorflow

22

23

LSTM – Long Short Term Memory

Resolver o problema da dependência longa



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

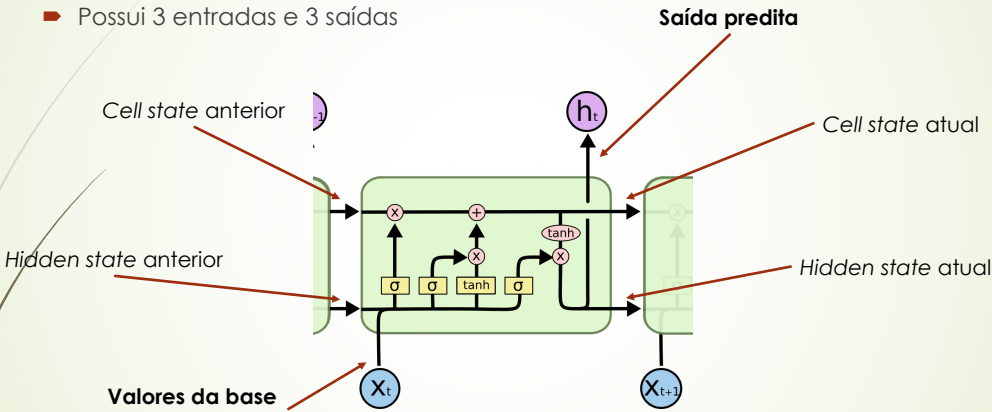
Tensorflow

23

24

LSTM – Long Short Term Memory

Possui 3 entradas e 3 saídas



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

24

25

LSTM – Long Short Term Memory

Possuem a capacidade de adicionar e remover informação

Por meio de portas / gates

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

25

26

LSTM – Long Short Term Memory

Estado da Célula (cell state)

Informação que flui através do tempo

Pode ser alterada através de portas

Camada Neural
Pesos e F. Ativação
Sigmóide (0 ~ 1)

0 – passa nada
1 – passa tudo

multiplicação

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

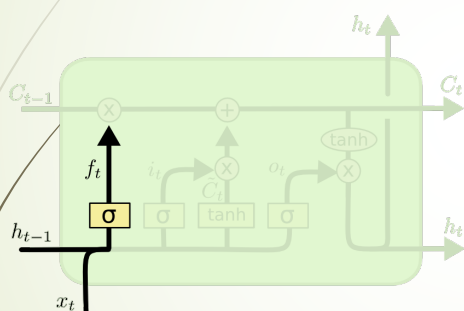
Tensorflow

26

27

LSTM – Long Short Term Memory

- Decidir o que passa pela *cell state*
- É feito com uma *forget gate*



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

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

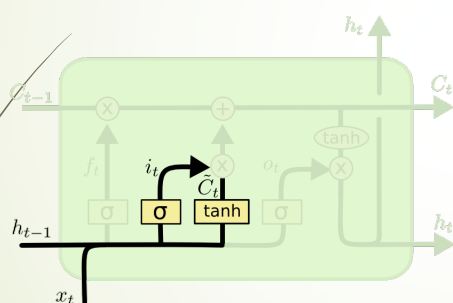
Tensorflow

27

28

LSTM – Long Short Term Memory

- Decidir qual informação nova é armazenada no *cell state*
 - Decidir quais valores **atualizar** : feito com uma *input gate layer*
 - Decidir os **novos** candidatos : feito com uma porta *tanh*



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

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

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

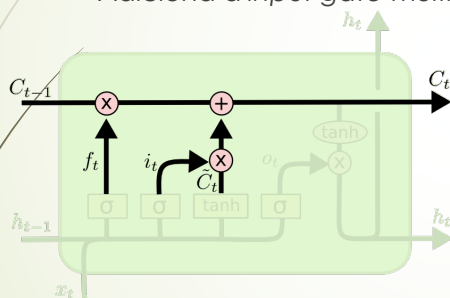
28

29

LSTM – Long Short Term Memory

- Atualizar o cell state

- Multiplica o estado anterior pela *forget gate*
- Adiciona a *input gate* multiplicado pelos novos candidatos



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

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

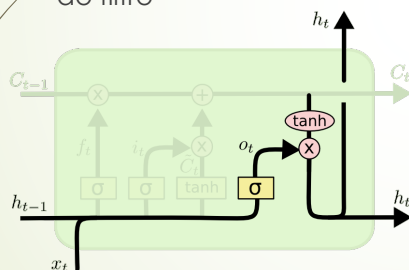
29

30

LSTM – Long Short Term Memory.

- Decidir qual a saída

- É baseada no cell state, mas filtrado por um sigmóide
- Usa um sigmóide (filtro) para decidir quais partes do cell state sairão
- Depois, cell state passa por um tanh (-1, 1) e multiplica pelo resultado do filtro



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

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

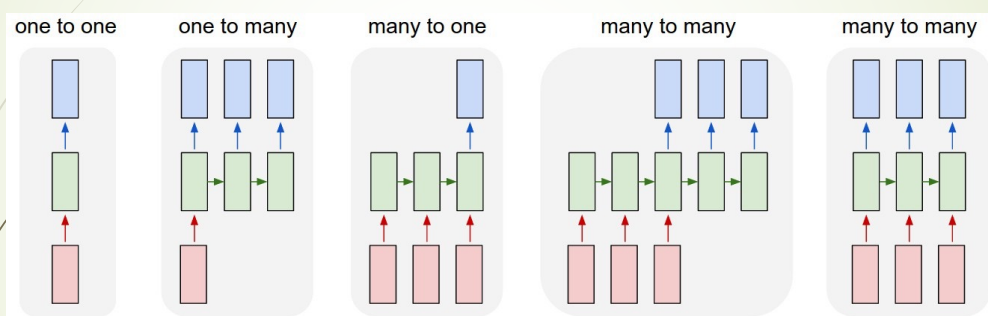
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

30

31

Tipos de Redes Neurais Recorrentes



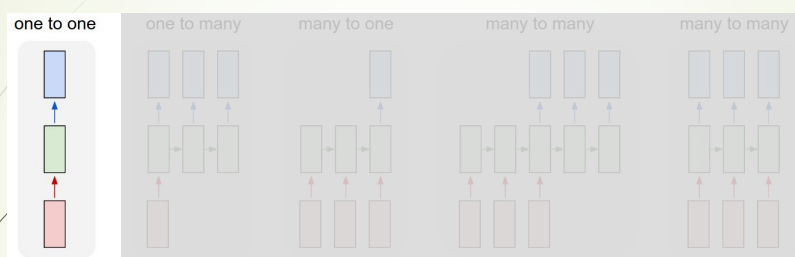
Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

31

32

Tipos de Redes Neurais Recorrentes



- Redes clássicas *feedforward*
- Não são recorrentes!
- Os dados são lidos uma vez, passados por várias épocas para treinamento e tem-se um resultado

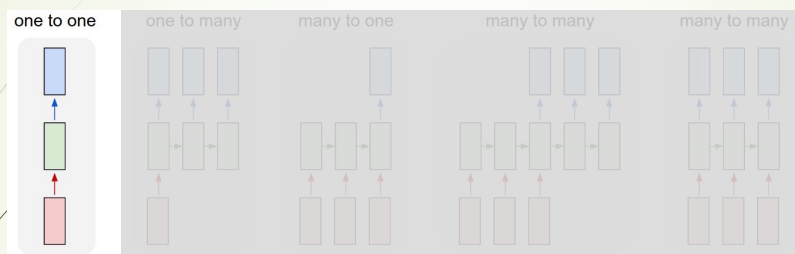
Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

32

33

Tipos de Redes Neurais Recorrentes



```
i = Input(shape=(T,))
x = Dense(K)(i)
model = Model(i, x)
```

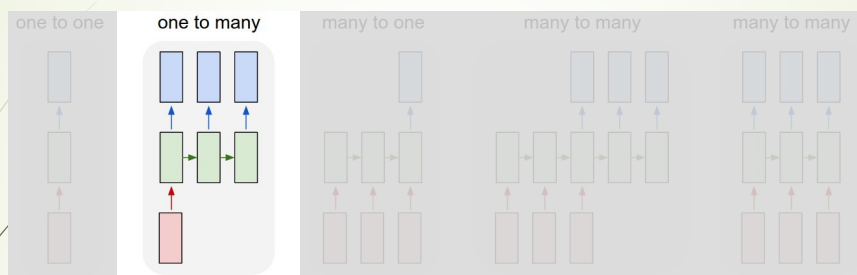
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

33

34

Tipos de Redes Neurais Recorrentes



- Os dados são lidos uma vez para as camadas ocultas
- A camada oculta é processada várias vezes e gera vários resultados
- Ex: descrição de imagens
- <https://ai.googleblog.com/2016/09/show-and-tell-image-captioning-open.html>

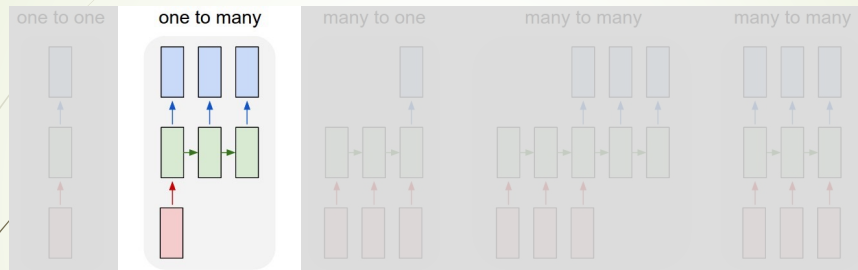
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

34

35

Tipos de Redes Neurais Recorrentes



```
i = Input(shape=(T, D))
x = RepeatVector(number_of_times)(i)
x = LSTM(M, return_sequences=True)(x)
x = Dense(K)(x)
model = Model(i, x)
```

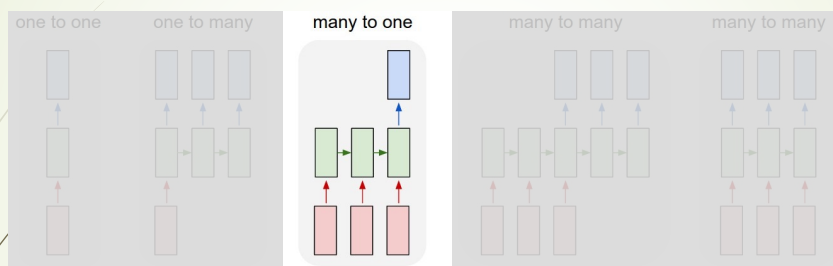
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

35

36

Tipos de Redes Neurais Recorrentes



- Lê-se várias entradas para gerar um resultado
- Ex. Análise de Sentimento, lê o texto todo e dá um resultado
- <https://matheusfacure.github.io/2017/05/26/deep-nlp-sentiment/>

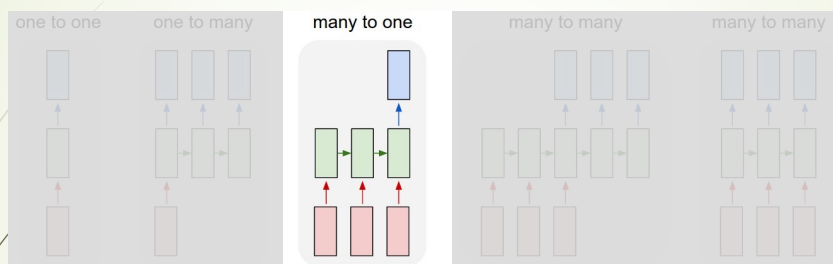
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

36

37

Tipos de Redes Neurais Recorrentes



```
i      = Input(shape=(T, D))
x      = LSTM(1)(i)
model = Model(i, x)
```

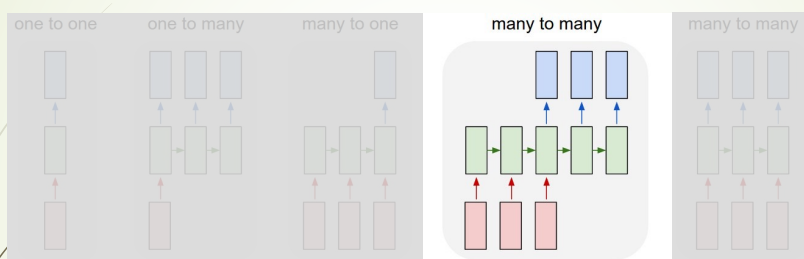
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

37

38

Tipos de Redes Neurais Recorrentes



- Várias entradas com várias saídas, com uma defasagem na primeira saída
- Ex. Tradução automática
- https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/recurrent_neural_networks/machine-translation-using-rnn.html

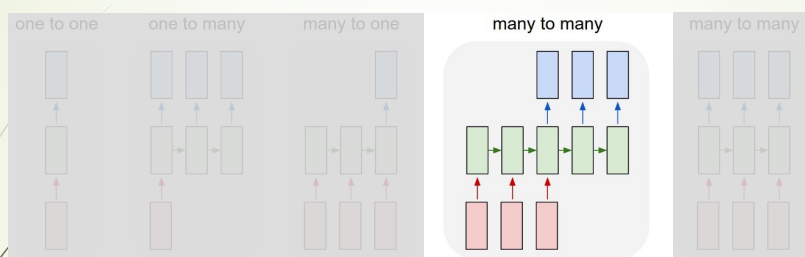
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

38

39

Tipos de Redes Neurais Recorrentes



```
i      = Input(shape=(T, D))
x      = LSTM(1, return_sequences=True)(i)
x      = Dense(K)(x)
model = Model(i, x)
```

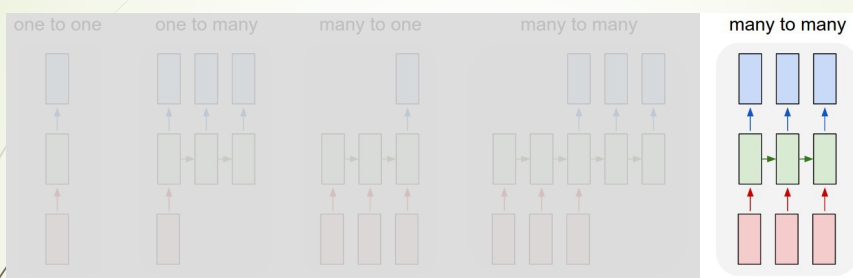
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

39

40

Tipos de Redes Neurais Recorrentes



- Várias entradas gerando várias saídas
- Ex. Previsões usando séries temporais

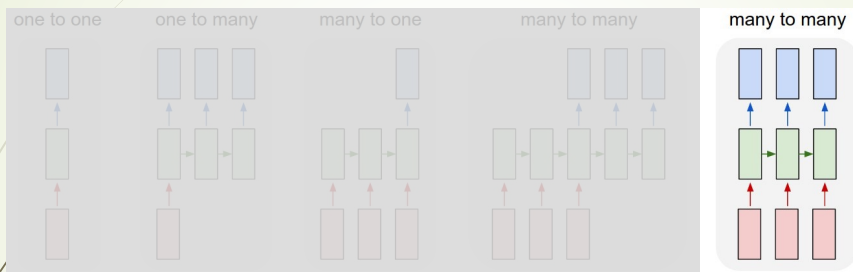
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

40

41

Tipos de Redes Neurais Recorrentes.



```
i      = Input (shape=(T, D))
x      = LSTM(M) (i)
x      = Dense(K) (x)
model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

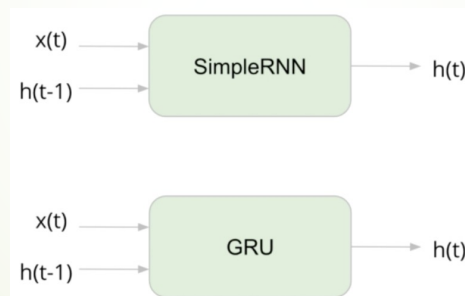
Tensorflow

41

42

GRU– Gated Recurrent Unit

- É um tipo de LSTM, parecido com SimpleRNN
- Combina a *forget gate* e *input gate* em uma porta *update gate*
- Combina o *cell state* com *hidden state*
- A saída é $h(t)$



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

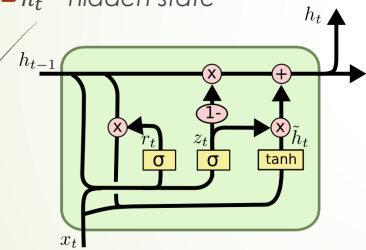
Tensorflow

42

43

GRU– Gated Recurrent Unit.

- z_t – Indica se o valor de h_{t-1} deve ser mantido ou atualizado
- r_t – Indica se deve-se lembrar ou esquecer h_{t-1}
- \tilde{h}_t – hidden state candidato
- h_t – hidden state



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

z_t – vetor update
 r_t – vetor reset
 \tilde{h}_t – hidden state candidato
 h_t – hidden state

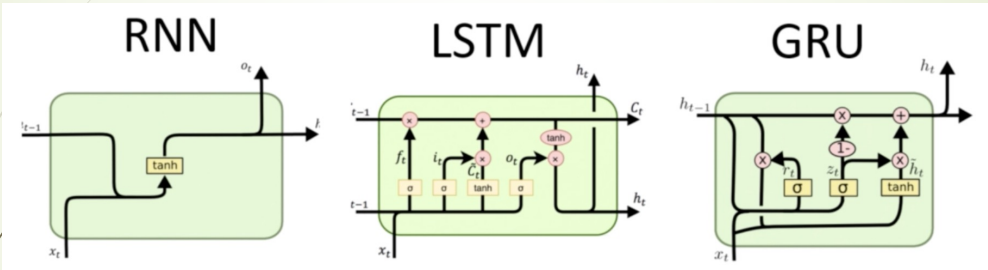
Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

43

44

Comparativo.



Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

44

45

LSTM

- ▀ LSTM no TensorFlow/Keras
- ▀ Valores
 - ▀ T – tamanho da sequência a ser entrada na rede
 - ▀ D – número de atributos em cada dado
 - ▀ M – número de unidades escondidas
 - ▀ K – número de nodos de saída

```
i      = Input (shape=(T, D))
x      = LSTM(M) (i)
x      = Dense(K) (x)
```

```
model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

45

46

GRU..

- ▀ GRU no TensorFlow/Keras
- ▀ Valores
 - ▀ T – tamanho da sequência a ser entrada na rede
 - ▀ D – número de atributos em cada dado
 - ▀ M – número de unidades escondidas
 - ▀ K – número de nodos de saída

```
i      = Input (shape=(T, D))
x      = GRU(M) (i)
x      = Dense(K) (x)
```

```
model = Model(i, x)
```

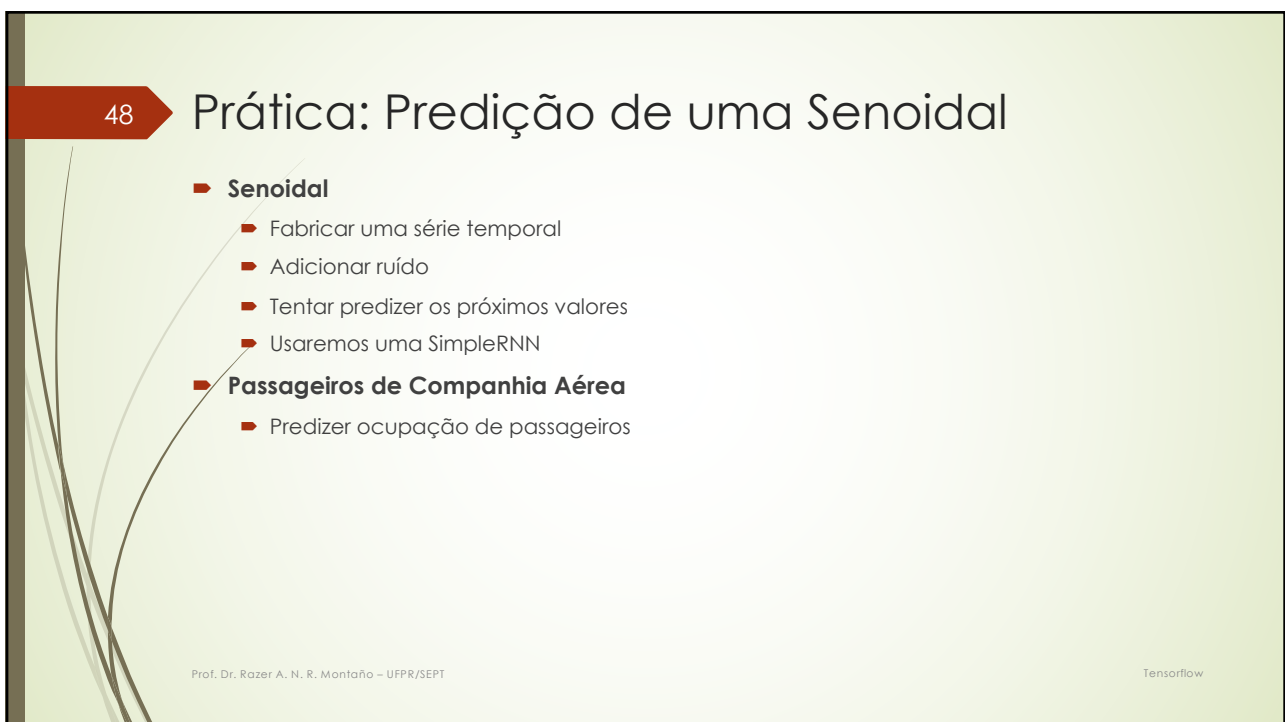
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

46



47



48

49

Senoidal

► Importações

```
import tensorflow as tf
tf.__version__

from tensorflow.keras.layers import Input, SimpleRNN, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD, Adam

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

49

50

Senoidal

► Criação dos dados

```
series = np.sin(0.1*np.arange(200)) + np.random.randn(200)*0.1

plt.plot(series)
plt.show()
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

50

51

Senoidal

- Criação do dataset (dimensões corretas – $N \times T \times D$)

```
T = 10 # tamanho da janela
D = 1 # quantidade de dados
X = []
Y = []

for t in range(len(series)-T):
    x = series[t:t+T]
    X.append(x)
    y = series[t+T]
    Y.append(y)

X = np.array(X).reshape(-1, T, 1) # Fica  $N \times T \times D$ 
Y = np.array(Y)
N = len(X)

print("X.shape: ", X.shape, " Y.shape: ", Y.shape)
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

51

52

Senoidal

- Criação do Modelo

```
i = Input(shape=(T, 1))
x = SimpleRNN(5, activation="relu")(i)
x = Dense(1)(x)

model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

52

53

Senoidal

► Compilação do Modelo

```
model.compile(  
    loss="mse",  
    optimizer=Adam(lr=0.1)  
)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

53

54

Senoidal

► Ajuste do Modelo

```
r = model.fit(  
    X[:-N//2], Y[:-N//2],  
    epochs=80,  
    validation_data=(X[-N//2:], Y[-N//2:])  
)
```

► Usa

- Metade da base para treino : $X[:-N//2]$, $Y[:-N//2]$
- Metade para teste : $X[-N//2:]$, $Y[-N//2:]$

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

54

55

Senoidal

- Plotar a função de perda

```
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

55

56

Senoidal

- Predições : 1-step

```
validation_target = Y[-N//2:]
validation_predictions = []

i = -N//2

while len(validation_predictions) < len(validation_target):
    p = model.predict( X[i].reshape(1, -1, 1)) [0, 0]
    i += 1

    validation_predictions.append(p)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

56

57

Senoidal

- Plotar Observado x Predito

```
plt.plot(validation_target, label="Observado")  
plt.plot(validation_predictions, label="Predito")  
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

57

58

Senoidal.

- Plotar Observado x Predito

```
plt.plot(validation_target, label="Observado")  
plt.plot(validation_predictions, label="Predito")  
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

58

59

EXERCÍCIO.

- ▶ Executar o exercício de senoidal.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

59

60

Passageiros

- ▶ Importações

```
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.layers import Input, SimpleRNN, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD, Adam

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

tf.__version__
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

60

61

Passageiros

➤ Carga da base

```
!wget http://www.razer.net.br/datasets/airline-
passengers.csv

df = pd.read_csv("airline-passengers.csv", usecols=[1])

df.head()

plt.plot(df)
plt.show()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

61

62

Passageiros

➤ Pré-processamento

```
series = df.values
series = series.astype('float32')

scaler = MinMaxScaler(feature_range=(0, 1))
series = scaler.fit_transform(series)

# tamanho da base de treino - 67% do total
train_size = int(len(series) * 0.67)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

62

63

Passageiros

➤ Conversão da série em dataset

```
T = 10
D = 1
X = []
Y = []

for t in range(len(series)-T):
    x = series[t:t+T]
    X.append(x)
    y = series[t+T]
    Y.append(y)

X = np.array(X).reshape(-1, T, 1) # Fica N x T x D
Y = np.array(Y)
N = len(X)

print("X.shape: ", X.shape, " Y.shape: ", Y.shape)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

63

64

Passageiros

➤ Criação do modelo

```
i = Input(shape=(T, 1))

x = SimpleRNN(5, activation=None)(i)
# x = SimpleRNN(5, activation="relu")(i)
# x = SimpleRNN(5)(i) # tanh por default

x = Dense(1)(x)

model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

64

65

Passageiros

► Compilação e ajuste do modelo

```
model.compile(  
    loss="mse",  
    optimizer=Adam(learning_rate=0.1)  
)  
  
r = model.fit(  
    X[:train_size], Y[:train_size],  
    epochs=80,  
    validation_data=(X[-train_size:], Y[-train_size:])  
)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

65

66

Passageiros

► Plotar Função de Perda

```
plt.plot(r.history["loss"], label="loss")  
plt.plot(r.history["val_loss"], label="val_loss")  
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

66

67

Passageiros

1-Step

```
validation_target = Y[-train_size:]
validation_predictions = []

i = -train_size

while len(validation_predictions) < len(validation_target):
    p = model.predict( X[i].reshape(1, -1, 1))[0, 0]
    i += 1

    validation_predictions.append(p)

plt.plot(validation_target, label="Observado")
plt.plot(validation_predictions, label="Predito")
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

67

68

Passageiros.

Multi-Step

```
validation_target = Y[-train_size:]
validation_predictions = []

last_x = X[-train_size]

while len(validation_predictions) < len(validation_target):
    p = model.predict( last_x.reshape(1, -1, 1))[0, 0]

    validation_predictions.append(p)

    last_x = np.roll(last_x, -1)
    last_x[-1] = p

plt.plot(validation_target, label="Observado")
plt.plot(validation_predictions, label="Predito")
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

68



69

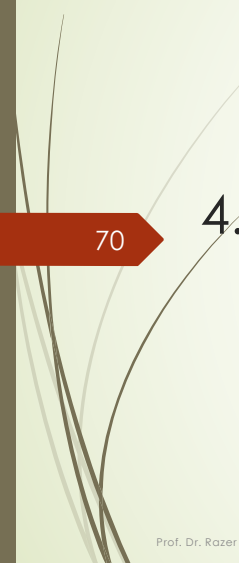
EXERCÍCIO.

- ▶ Executar o exercício de passageiros.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

69



70

4.2 Classificação de Textos com RNN

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

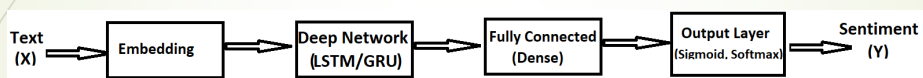
Tensorflow

70

71

Classificação de Textos com RNN

Arquitetura



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

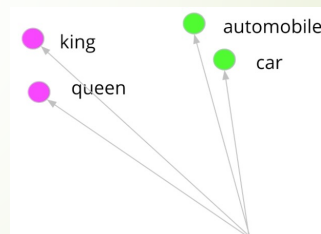
Tensorflow

71

72

Classificação de Textos com RNN

- Como codificar uma frase para entrar em uma Rede Neural?
- Tokenização (**tokenizer**)
 - Cada palavra um número:
 - "I like Java" -> ["I", "like", "Java"] -> [10, 8, 200]
- **Embedding Matrix**
 - Usa os números para indexar uma matriz contendo os vetores das palavras
 - [10, 8, 200] -> [[0.2, -1.4], [2.1, 0.5], [0.4, 1.2]]
 - Array de tamanho T => Matriz de Tamanho T x D
 - Os pesos indicam as similaridades entre as palavras



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

72

73

Classificação de Textos com RNN

- A matriz de Embeddings pode ser treinada, com descida de gradiente
- Pode-se usar uma matriz pré-treinada
 - Com outros algoritmos como word2vec, GloVe
- Para processar textos, é necessária a **camada Embedding**



Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

73

74

Classificação de Textos com RNN

- Pré-processamentos
 - Tokenização (**tokenizer**) : Para transformar os textos em inteiros
 - Usa a classe **Tokenizer**
 - Preenchimento (**padding**) : Para transformar as sequências de palavras (sentenças) em sequências do mesmo tamanho
 - Para se obter uma matriz $N \times T$
 - N – número de amostras
 - T – tamanho das sentenças
 - Função **pad_sequences()**

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

74

75

Classificação de Textos com RNN..

- Treinamento e predições ocorrem normalmente
 - Input
 - Embedding
 - LSTM
 - GlobalMaxPooling1D (opcional)
 - Dense
- Pode-se usar uma camada de GlobalMaxPooling1D
 - Caso `return_sequences=True` na camada LSTM
 - 1D – dados temporais
 - 2D – dados espaciais
 - 3D – dados espaciais ou espaço-temporais

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

75

76

LABORATÓRIO

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

76

77

Prática: Classificação de Reviews IMDB

IMDB

- Base de reviews do IMDB
- Cada avaliação é uma lista de palavras
- As palavras já estão codificadas
 - Não precisa de tokenizer
- Detectar se é uma avaliação positiva ou negativa
- Avaliações possuem tamanhos diferentes
 - Precisa padronizar para entrada na rede
- Sentimento: 0 – Negativo / 1 - Positivo

```
x_train.shape: (25000,)
y_train.shape: (25000,)
x_test.shape: (25000,)
y_test.shape: (25000,)
```

```
array([list([1, 14, 22, 16, 43,
            list([1, 194, 1153, 194,
            list([1, 14, 47, 8, 30, 3
            ...,
            list([1, 11, 6, 230, 245,
            list([1, 1446, 7079, 69,
            list([1, 17, 6, 194, 337,
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

77

78

IMDB

Importações

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.datasets import imdb

tf.__version__
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

78

79

IMDB

Carga da Base IMDB

```
num_words = 20000 # número de palavras
maxlen = 200      # máximo palavras no review

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_words)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

79

80

IMDB

Algumas verificações

```
x_train.shape
```

```
(25000,)
```

```
x_train
```

```
array([list([1, 14, 22, 16, 43, 530, 973, 1622,
            list([1, 194, 1153, 194, 8255, 78, 228, 5
            list([1, 14, 47, 8, 30, 31, 7, 4, 249, 10]
```

```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

80

81

IMDB

- Algumas verificações – tamanhos dos reviews

```
print(len(x_train[0]), len(x_train[1]))
```

```
↳ 218 189
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

81

82

IMDB

- Transforma as reviews para ter o mesmo tamanho

```
x_train = pad_sequences(x_train, maxlen=maxlen)
print(len(x_train[0]), len(x_train[1]))
```

```
↳ 200 200
```

```
x_test = pad_sequences(x_test, maxlen=maxlen)
print(len(x_test[0]), len(x_test[1]))
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

82

83

IMDB

➤ Construir o Modelo

```
# Construir a RNN
# recebe como entrada maxlen palavras
i = Input(shape=(x_train.shape[1], ))

# com textos precisa da camada de
# - representação numérica das palavras
# num_words palavras e 128 características para representar
# cada palavra (colunas)
# Não tem relação com a quantidade de unidades da LSTM
x = Embedding(input_dim=num_words, output_dim=128)(i)
x = LSTM(units=128, activation="tanh")(x)
x = Dense(units=1, activation="sigmoid")(x)

model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

83

84

IMDB

➤ Compilar o Modelo

```
# para RNN melhor rmsprop
model.compile(optimizer="rmsprop", loss="binary_crossentropy",
              metrics=["accuracy"])

model.summary()
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

84

85

IMDB

Treinar o Modelo

```
epochs = 10  
  
r = model.fit(x_train, y_train, epochs=epochs, batch_size=128)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

85

86

IMDB

Gráfico da loss-function e acurácia

```
plt.plot( r.history["loss"], label="loss")  
plt.xlabel("Épocas")  
plt.ylabel("loss")  
plt.xticks(np.arange(0, epochs, step=1))  
plt.legend()
```

```
plt.plot( r.history["accuracy"], label="accuracy")  
plt.xlabel("Épocas")  
plt.ylabel("Acurácia")  
plt.xticks(np.arange(0, epochs, step=1))  
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

86

87

IMDB

➤ Avaliação na base de Teste

```
test_loss, test_accuracy = model.evaluate(x_test, y_test)
```

```
print(test_accuracy)
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

87

88

IMDB.

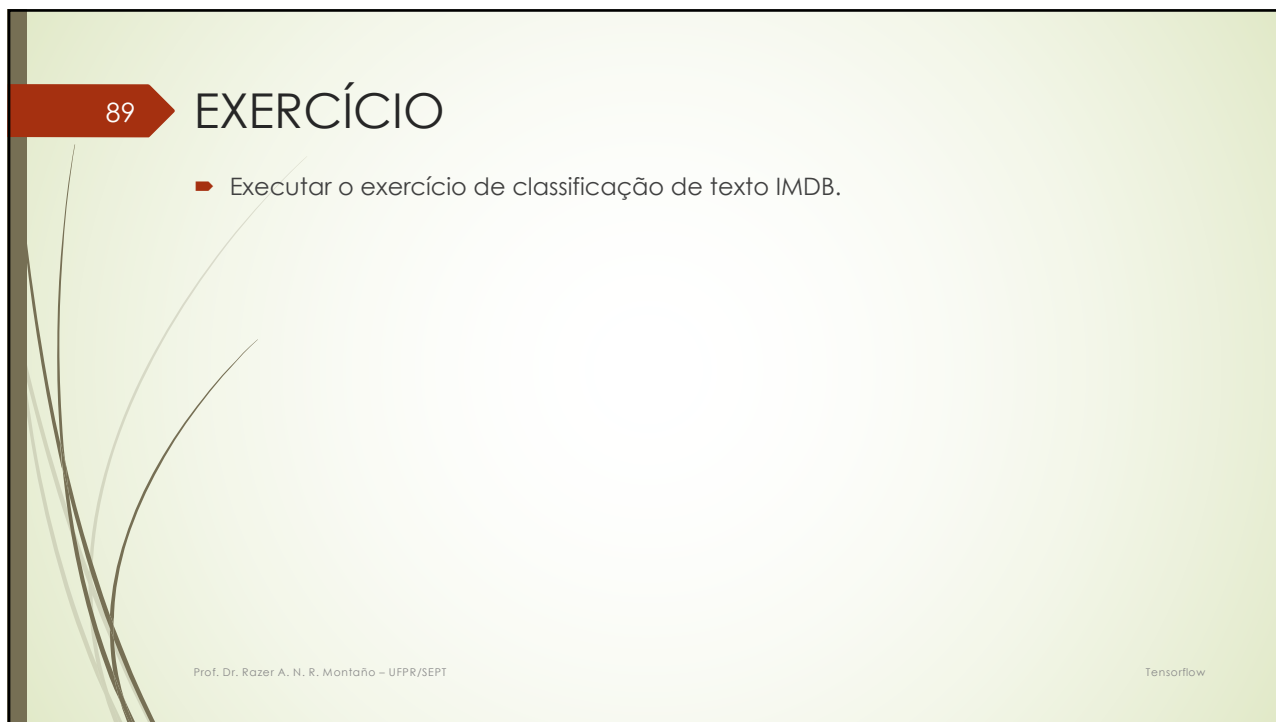
➤ Efetuar uma predição

```
texto = [[4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766]]  
entrada = pad_sequences(texto, 200)  
sent = np.round(model.predict(entrada))  
  
print( "Positivo" if sent==1 else "Negativo" )
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

88



89


EXERCÍCIO

- ▶ Executar o exercício de classificação de texto IMDB.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

89



90

LABORATÓRIO

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

90

91

Prática: Classificação de E-mails - SPAM

➤ SPAM

- Base de e-mails
- Precisa fazer a tokenização

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

91

92

SPAM

➤ Importações

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.model_selection import train_test_split

from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.layers import GlobalMaxPooling1D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

92

93

SPAM

- Obter a base e carregar num DataFrame

```
!wget http://www.razer.net.br/datasets/spam.csv

df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()

df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam" : 1})
y = df["b_labels"].values
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

93

94

SPAM

- Separar a base em Treino/Teste

```
x_train, x_test, y_train, y_test = train_test_split(df["data"],
y, test_size=0.33)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

94

95

SPAM

Tokenização

```
# Número máximo de palavras para considerar
# São consideradas as mais frequentes, as demais são
# ignoradas
num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)

word2index = tokenizer.word_index
V = len(word2index)
print("%s tokens" % V)
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

95

96

SPAM

Padding

```
data_train = pad_sequences(sequences_train)

T = data_train.shape[1] # tamanho da sequência

data_test = pad_sequences(sequences_test, maxlen=T)

print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

96

97

SPAM

➤ Criação do Modelo

```
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
M = 15 # tamanho do hidden state

i = Input(shape=(T,))
x = Embedding(V+1, D)(i)
# x = LSTM(M)(x) # tirar o GlobalMaxPooling1D()
x = LSTM(M, return_sequences=True)(x)

x = GlobalMaxPooling1D()(x)
x = Dense(1, activation="sigmoid")(x)

model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

97

98

SPAM

➤ Compilação e Treino do Modelo

```
model.compile(
    loss="binary_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)

epochs = 10
r = model.fit(
    data_train,
    y_train,
    epochs=epochs,
    validation_data=(data_test, y_test))
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

98

99

SPAM

- Plotar a função de perda

```
plt.plot( r.history["loss"], label="loss")
plt.plot( r.history["val_loss"], label="val_loss")
plt.xlabel("Épocas")
plt.ylabel("loss")
plt.xticks(np.arange(0, epochs, step=1))
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

99

100

SPAM

- Plotar a acurácia

```
plt.plot( r.history["accuracy"], label="accuracy")
plt.plot( r.history["val_accuracy"], label="val_accuracy")
plt.xlabel("Épocas")
plt.ylabel("Acurácia")
plt.xticks(np.arange(0, epochs, step=1))
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

100

101

SPAM

- Predição de um novo texto

```
texto      = "Before I introduce myself <escreva um texto aqui> "  
  
seq_texto  = tokenizer.texts_to_sequences([texto])  
data_texto = pad_sequences(seq_texto, maxlen=T)  
  
pred       = model.predict(data_texto)  
  
print(pred)  
print ("SPAM" if pred >= 0.5 else "OK")
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

101

102

EXERCÍCIO

- Executar o exercício de classificação de SPAM.

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

102



103

A slide with a light green background and a dark green vertical bar on the left. The title "Prática: Geração de Textos com RNN" is written in large, bold, black capital letters. A red arrow points to the number "104" on the left side. Below the title, there is a list of bullet points: "Texto de Shakespeare (retirado do site oficial do TensorFlow)", "Entrada: 'Shakespear'", "Predição: 'e'", and "A produção do texto pode ser feita chamando-se o modelo repetidamente". Below the list, there is a red URL: https://www.tensorflow.org/tutorials/text/text_generation. At the bottom, there is a small text "Prof. Dr. Razer A. N. R. Montão - UFPR/SEPT" and a "Tensorflow" logo.

104

105

Geração de Texto

■ Importações

```
import tensorflow as tf
import numpy as np
import os
import time
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

105

106

Geração de Texto

■ Carga do texto para treino

```
path_to_file = tf.keras.utils.get_file('shakespeare.txt',
'https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.txt')

# Leitura do texto
text = open(path_to_file, 'rb').read().decode(encoding='utf-8')

# Tamanho do texto em número de caracteres
print(f'Tamanho do texto: {len(text)} caracteres')

# Primeiros 250 caracteres do texto
print(text[:250])

# Caracteres únicos
vocab = sorted(set(text))
print(f'{len(vocab)} caracteres únicos')
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

106

107

Geração de Texto

Processamento do texto

```
# Processamento do texto

# Converte um caractere em um ID único
ids_from_chars = tf.keras.layers.StringLookup(vocabulary=list(vocab), mask_token=None)
# Faz o contrário, converte os IDs em caracteres
chars_from_ids =
tf.keras.layers.StringLookup(vocabulary=ids_from_chars.get_vocabulary(), invert=True,
mask_token=None)

# Função onde, dado uma lista de IDs, gera o texto
def text_from_ids(ids):
    return tf.strings.reduce_join(chars_from_ids(ids), axis=-1)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

107

108

Geração de Texto

Gerar Base de Treino

```
# Gerar base de treino
# Exemplo : Para a palavra "Hello"
# Suponha seq_length = 4
# Então: Entrada "Hell" e Saida "ello"
# Tem que dividir o texto em pedaços de tamanho seq_length+1

# from_tensor_slices - cria um dataset com os dados
all_ids = ids_from_chars(tf.strings.unicode_split(text, 'UTF-8'))
ids_dataset = tf.data.Dataset.from_tensor_slices(all_ids)

seq_length = 100
examples_per_epoch = len(text) // (seq_length+1)

# Converte as sequências no tamanho desejado : seq_length+1
sequences = ids_dataset.batch(seq_length+1, drop_remainder=True)

for seq in sequences.take(5):
    print(text_from_ids(seq).numpy())
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

108

109

Geração de Texto

► Gerar Base de Treino

```
# Função onde, dado uma sequência "Hello", gera entrada e saída: "Hell" e "ello"
def split_input_target(sequence):
    input_text = sequence[:-1]
    target_text = sequence[1:]
    return input_text, target_text

# dataset contém as sequências contendo entrada e saída
dataset = sequences.map(split_input_target)

# Criar lotes de treinamento

# Batch size
BATCH_SIZE = 64

# Tamanho do buffer para randomizar o dataset
BUFFER_SIZE = 10000
dataset = (
    dataset
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE, drop_remainder=True)
    .prefetch(tf.data.experimental.AUTOTUNE))
```

109

110

Geração de Texto

► Construção do modelo

```
# Construir o modelo

# Tamanho do vocabulário em número de caracteres
vocab_size = len(vocab)

# Dimensão do Embedding
embedding_dim = 256

# Número de unidades RNN
rnn_units = 1024
```

110

111

Geração de Texto

► Construção do modelo

```
# Classe que gera o modelo: Embedding -> GRU -> Dense
class MyModel(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, rnn_units):
        super().__init__(self)
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(rnn_units,
                                       return_sequences=True,
                                       return_state=True)
        self.dense = tf.keras.layers.Dense(vocab_size)

    def call(self, inputs, states=None, return_state=False, training=False):
        x = inputs
        x = self.embedding(x, training=training)
        if states is None:
            states = self.gru.get_initial_state(x)
        x, states = self.gru(x, initial_state=states, training=training)
        x = self.dense(x, training=training)

        if return_state:
            return x, states
        else:
            return x
```

Tensorflow

111

112

Geração de Texto

► Construção do modelo

```
# Criação do modelo
model = MyModel(
    # Be sure the vocabulary size matches the `StringLookup` layers.
    vocab_size=len(ids_from_chars.get_vocabulary()),
    embedding_dim=embedding_dim,
    rnn_units=rnn_units)
```

Prof. Dr. Razer A. N. R. Montañó - UFPR/SEPT

Tensorflow

112

113

Geração de Texto

► Compilação do Modelo

- Otimizador: Adam
- *Sparse Categorical Cross-Entropy*
- **from_logits**: Se as predições são em logits. Por default são probabilidades
 - Logits são valores brutos, não normalizados, usados como entrada em uma softmax
- Usado pois a camada Densa não tem função de ativação

$$L_{CE} = - \sum_{i=1}^n t_i \log(p_i), \text{ for } n \text{ classes,}$$

```
# Função de perda é sparse_categorical_crossentropy
# Modelo retorna Logits, sinaliza from_logits
loss = tf.losses.SparseCategoricalCrossentropy(from_logits=True)

# Compila o modelo
model.compile(optimizer='adam', loss=loss)

# Treinar
EPOCHS = 20
history = model.fit(dataset, epochs=EPOCHS)
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

113

114

Geração de Texto

► Classe para Geração – um passo

```
class OneStep(tf.keras.Model):
    def __init__(self, model, chars_from_ids, ids_from_chars, temperature=1.0):
        super().__init__()
        self.temperature = temperature
        self.model = model
        self.chars_from_ids = chars_from_ids
        self.ids_from_chars = ids_from_chars

    # Create a mask to prevent "[UNK]" from being generated.
    skip_ids = self.ids_from_chars(['[UNK]'][:, None])
    sparse_mask = tf.SparseTensor(
        # Put a -inf at each bad index.
        values=[-float('inf')] * len(skip_ids),
        indices=skip_ids,
        # Match the shape to the vocabulary
        dense_shape=[len(ids_from_chars.get_vocabulary())])
    self.prediction_mask = tf.sparse.to_dense(sparse_mask)
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

114

115

Geração de Texto

► Classe para Geração – um passo

```
@tf.function
def generate_one_step(self, inputs, states=None):
    # Convert strings to token IDs.
    input_chars = tf.strings.unicode_split(inputs, 'UTF-8')
    input_ids = self.ids_from_chars(input_chars).to_tensor()

    # Run the model.
    # predicted_logits.shape is [batch, char, next_char_logits]
    predicted_logits, states = self.model(inputs=input_ids, states=states,
                                          return_state=True)

    # Only use the last prediction.
    predicted_logits = predicted_logits[:, -1, :]
    predicted_logits = predicted_logits/self.temperature
    # Apply the prediction mask: prevent "[UNK]" from being generated.
    predicted_logits = predicted_logits + self.prediction_mask

    # Sample the output logits to generate token IDs.
    predicted_ids = tf.random.categorical(predicted_logits, num_samples=1)
    predicted_ids = tf.squeeze(predicted_ids, axis=-1)

    # Convert from token ids to characters
    predicted_chars = self.chars_from_ids(predicted_ids)

    # Return the characters and model state.
    return predicted_chars, states
```

Tensorflow

115

116

Geração de Texto.

► Geração – um passo

```
one_step_model = OneStep(model, chars_from_ids, ids_from_chars)

# Executar em um laço para gera o texto
start = time.time()
states = None
next_char = tf.constant(['ROMEO:'])
result = [next_char]

for n in range(1000):
    next_char, states = one_step_model.generate_one_step(next_char, states=states)
    result.append(next_char)

result = tf.strings.join(result)
end = time.time()
print(result[0].numpy().decode('utf-8'), '\n\n' + '_'*80)
print('\nRun time:', end - start)
```

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

116



117

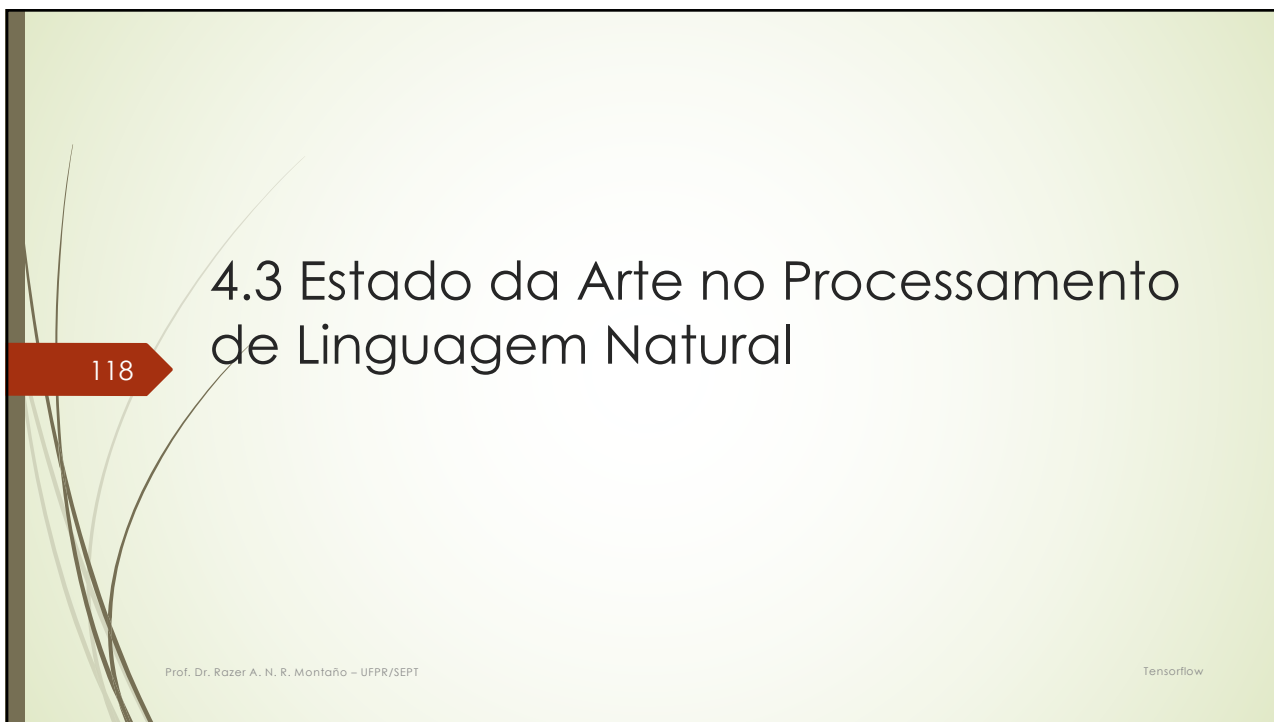
EXERCÍCIO..

- ▶ Executar o exercício de geração de texto.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

117



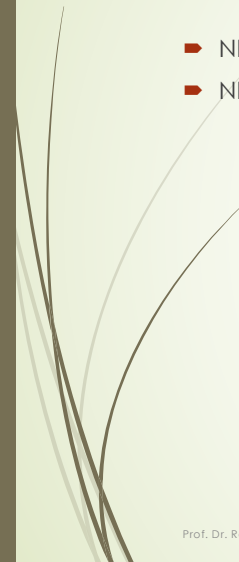
118

4.3 Estado da Arte no Processamento de Linguagem Natural

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

118




119 NLP

- NLU : Natural Language Understanding
- NLG : Natural Language Generation

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

119



120 Transformers

- Arquitetura de rede neural
 - Tratar sequência ordenada de dados
- Usadas para tarefas de tradução:
 - Reconhecimento de fala
 - Texto
- Não trata os dados de forma sequencial
 - Primeiro o início da sequência e só depois o final
 - Portanto, treinamento pode ser altamente paralelizado
- Substitui LSTM
- Deu origem a modelos pré-treinados: BERT, GPT-2, XLNet

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

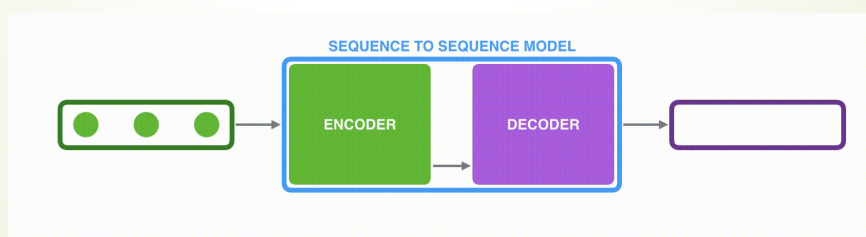
120

121

Arquitetura Encoder-Decoder

Arquitetura Encoder-Decoder

- Encoder lê a entrada, transforma em vetores de embeddings
- Transforma a entrada em um vetor de contexto
- Decoder gera a saída a partir do vetor de contexto
- Encoder e Decoder são RNNs



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

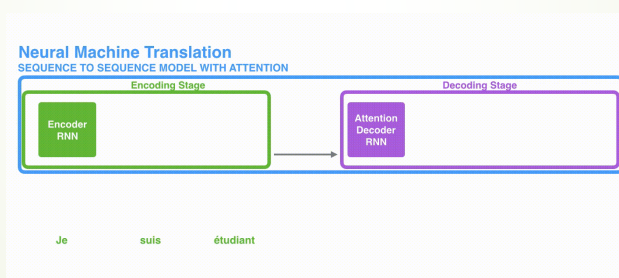
Tensorflow

121

122

Mecanismo de Atenção

- Contornar a limitação do tamanho fixo da entrada
- Criado para melhorar o desempenho da arquitetura ENCODER/DECODER para tradução automática
- Permite o DECODER usar partes mais relevantes da entrada, de forma flexível, com as partes mais relevantes possuindo um peso maior



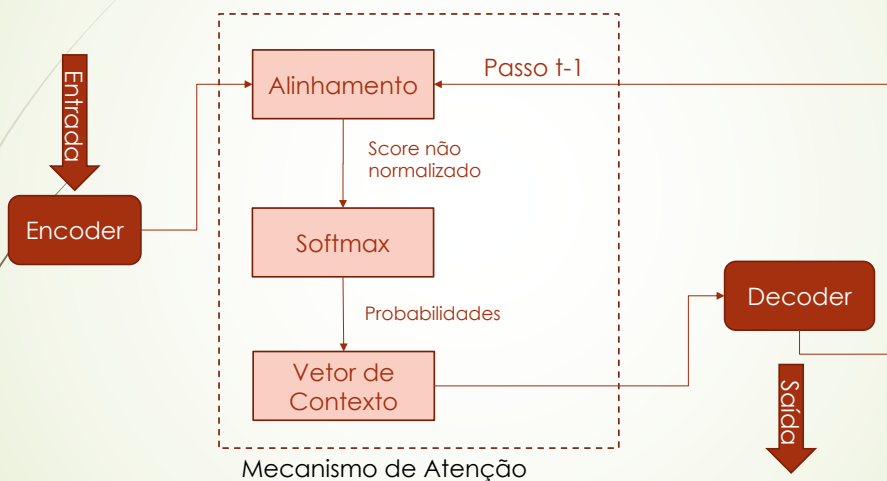
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

122

123

Mecanismo de Atenção



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

123

124

Transformers

- Revolucionou o uso da atenção
- Dispensa a recorrência (RNNs) e convoluções
- Usa um mecanismo de **auto atenção**
- **Auto Atenção**
 - Como não tem recorrência, adiciona um dado posicional
 - A sequência é representada pela relação entre palavras na mesma sentença
 - Isso gera um vetor de contexto (*embeddings contextualizados*)

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

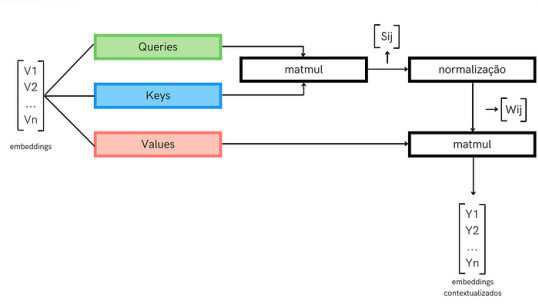
Tensorflow

124

125

Auto Atenção

- Cada elemento do vetor de Embedding (Q - Queries) é multiplicado pelos embeddings das outras palavras (K - Keys)
- Esses valores são normalizados e multiplicados pelos embeddings originais (V - Values)
- Esse é o vetor contextualizado



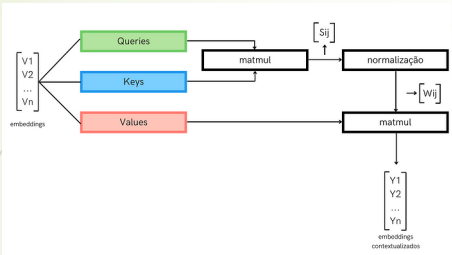
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

125

126

Auto Atenção



$$\begin{aligned} V_1 \cdot V_1 &= S_{11} \rightarrow W_{11} \\ V_1 \cdot V_2 &= S_{12} \rightarrow W_{12} \\ V_1 \cdot V_3 &= S_{13} \rightarrow W_{13} \\ &\dots \end{aligned}$$

$$V_1 \cdot V_{10} = S_{110} \rightarrow W_{110}$$

Query Keys

$$W_{11} \cdot V_1 + W_{12} \cdot V_2 + \dots + W_{110} \cdot V_{10} = Y_1$$

Values

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

126

127

Auto Atenção

Os parâmetros de treinamento são adicionados em cada etapa (M_i)

$V_1 M_q \cdot V_1 M_k = S_{11} \rightarrow W_{11}$

$V_1 M_q \cdot V_2 M_k = S_{12} \rightarrow W_{12}$

$V_1 M_q \cdot V_3 M_k = S_{13} \rightarrow W_{13}$

\vdots

$V_1 M_q \cdot V_{10} M_k = S_{110} \rightarrow W_{110}$

Query

Keys

$W_{11} \cdot V_1 M_v + W_{12} \cdot V_2 M_v + \dots + W_{110} \cdot V_{10} M_v = Y_1$

Values

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

127

128

Auto Atenção

Para dar atenção a mais de uma palavra ao mesmo tempo, usam-se camadas paralelas

$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$

embeddings

Linear

Linear

Linear

h camadas lineares paralelas, inicializadas com pesos aleatórios diferentes

$\begin{bmatrix} S_{ij}^1 & \dots & S_{ij}^h \end{bmatrix}$

matmul

normalização

$\begin{bmatrix} W_{ij}^1 \\ \vdots \\ W_{ij}^h \end{bmatrix}$

matmul

$\begin{bmatrix} Y_1^1 & \dots & Y_1^h \\ Y_2^1 & \dots & Y_2^h \\ \vdots & \vdots & \vdots \\ Y_n^1 & \dots & Y_n^h \end{bmatrix}$

$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$

embeddings contextualizados

Concatenação e camada densa

$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

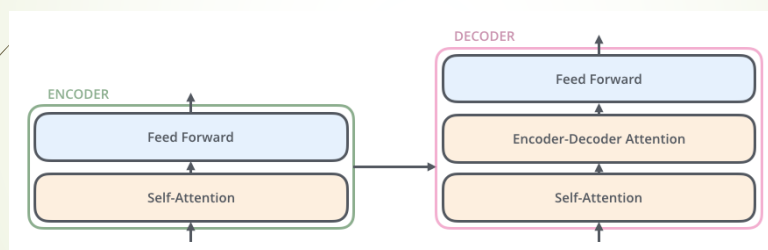
Tensorflow

128

129

Transformers

- Possuem uma camada de ENCODER e DECODER
 - Conectados entre si
 - Possui n encoders e n decoders



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

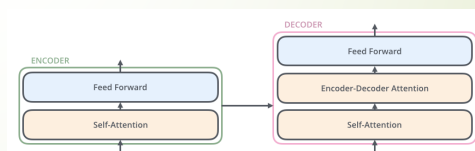
Tensorflow

129

130

Transformers

- Encoder
 - Camada de auto atenção
 - Camada de RN feed-forward
 - Ao terminar, envia a saída para o próximo encoder
- Decoder
 - Camada de mascaramento, para prever usando somente palavras já preditas anteriormente
 - Camada de auto atenção, que recebe as palavras anteriormente decodificadas e os vetores KEYS e VALUES do Encoder
 - Camada de RN feed-forward

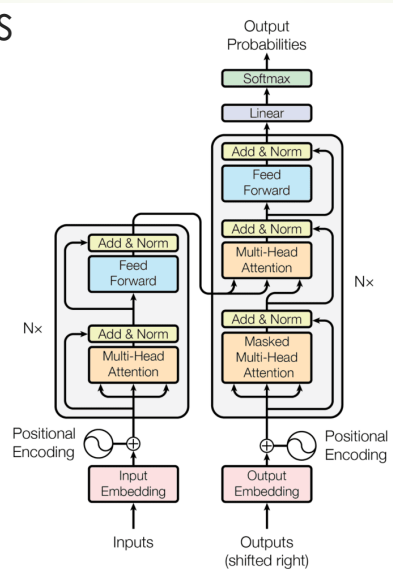


Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

130

131 Transformers



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

131

132 Exercício

- Implementar a tradução de texto usando Transformer no Tensorflow
- Seguir o tutorial: <https://www.tensorflow.org/text/tutorials/transformer?hl=pt-br>

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

132

133

BERT e GPT-2

- BERT: Bidirectional Encoder Representations From Transformers
 - Google
 - NLU – Natural Language Understanding
 - Usa palavras para frente e para trás para dar contexto (bidirecional)
 - Usado no motor de busca
 - Entender contextos
- GPT-2 (*Generative Pre-trained Transformer*)
 - Prediz a próxima palavra, dadas as palavras anteriores de um texto
 - Treinado com 40Gb de textos da Internet
 - Inicialmente não foi liberado ao público pelo possível uso incorreto, como gerador de fake news
 - <https://medium.com/ensina-ai/ensinando-português-ao-gpt-2-d4aa4aa29e1d>

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

133

134

GPT-3

- GPT-3 (*Generative Pre-trained Transformer*)
 - 2020: Sucessor do GPT-2
 - 175 bilhões de parâmetros treinados
 - Treinamento perto de 1 trilhão de palavras
 - Textos gerados MUITO próximos a de humanos
- Artigo original
 - <https://arxiv.org/abs/2005.14165>
- Artigo gerado pelo GPT-3
 - <https://maraoz.com/2020/07/18/openai-gpt3/>
- Chatbot baseado em GPT-3
 - <https://www.quickchat.ai/emerson>

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

134

135

GPT-3

- Exemplo
 - Geração de descrição de produtos
 - <https://www.copy.ai/>

PRODUCT/BRAND NAME (OPTIONAL)

Super Keyboard

DESCRIBE YOUR PRODUCT

a keyboard with artificial intelligence to predict words

56/400

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

135

136

GPT-3

Confused by the conventional QWERTY layout? The Super Keyboard can save you from the pains of learning new finger positions. It is a keyboard with artificial intelligence to predict words. Just type easy-to-combine symbols directly onto the keyboard, and it will determine your word for you! Never mistype again. The possibilities are endless...

A keyboard that "thinks" and predicts the words you're going to type next – by learning what you write, how you write it and how you like to write. The Super Keyboard will predict your next word, the name of a contact, or a symbol. It can be configured to support any language (

Introducing the AI-powered Super Keyboard, with the world's most advanced AI voice assistant that predicts your next words. The Super Keyboard integrates a complete set of smart features and personally understands you the more you use it. Enjoy an enjoyable experience typing key with the bigger key cover, smooth and quick Bluetooth connection, and type without looking thanks to the camera key light.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

136

137

ChatGPT

- Lançado em Nov/2022
 - 100 milhões de usuários por mês em Jan/2023
- Grande Modelo de Linguagem (LLM)
- Usa o modelo GPT-3.5
 - Um fine tuning do GPT 3
- Treinado com RLHF : Aprendizado por Reforço com Feedback Humano
- Características
 - Não acessa internet
 - Treinado com textos até 2021
 - Não tem compromisso com conteúdo

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

137

138

ChatGPT.



Escreva um texto de finalização de conteúdo da disciplina de Frameworks de IA, onde estudamos o TensorFlow. Agradeça o empenho dos alunos.



Nesta disciplina, tivemos a oportunidade de explorar um dos Frameworks de IA mais populares e poderosos da atualidade: o TensorFlow. Aprendemos sobre sua arquitetura, sua sintaxe e como usá-lo para construir modelos de aprendizado de máquina e redes neurais profundas.

Durante todo o curso, vocês se empenharam muito em aprender sobre o TensorFlow e aplicar seus conhecimentos em projetos práticos. Foi muito gratificante ver como cada um de vocês evoluiu ao longo do tempo e como os projetos finais demonstraram a habilidade e o comprometimento de todos.

A inteligência artificial é uma área em constante evolução, e o TensorFlow tem sido uma ferramenta fundamental para impulsionar a inovação nesse campo. Esperamos que, além de ter aprendido a usar essa poderosa ferramenta, vocês tenham compreendido a importância da inteligência artificial e do aprendizado de máquina em nossa sociedade.

Agradecemos pelo empenho de cada um de vocês, pelo interesse em aprender e pela contribuição para as discussões em sala de aula. Desejamos a todos muito sucesso em suas carreiras e que continuem se atualizando e se aprimorando no campo da inteligência artificial. Parabéns pela dedicação e pelo ótimo desempenho durante esta disciplina!

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

138

139

XLNet.

- Um modelo parecido com o BERT
 - Entendimento de Linguagem Natural
 - Olha o contexto das palavras: para frente/para trás
 - Como o GPT-2 e BERT
 - Melhor que o BERT em várias tarefas
- Usa contexto bidirecional
- Autoregressive language model
 - Usa o contexto, inclusive, do que já foi predito
- <https://arxiv.org/abs/1906.08237>

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

139

140

Jurassic-1..

- Da AI21 Labs
 - 2021
 - 178 bilhões de parâmetros
 - Geração de textos, sumarização de texto
 - Geração de código partir de um texto descritivo
 - Tradução de funções de uma linguagem para outra
 - Classificação de texto
- Links
 - Paper: https://uploads-ssl.webflow.com/60fd4503684b466578c0d307/61138924626a6981ee09caf6_jurassic_tech_paper.pdf
 - <https://www.ai21.com/blog/announcing-ai21-studio-and-jurassic-1>
 - Casos de Uso: <https://www.ai21.com/blog/ai21-studio-use-cases>

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

140