DEEPLEARNING

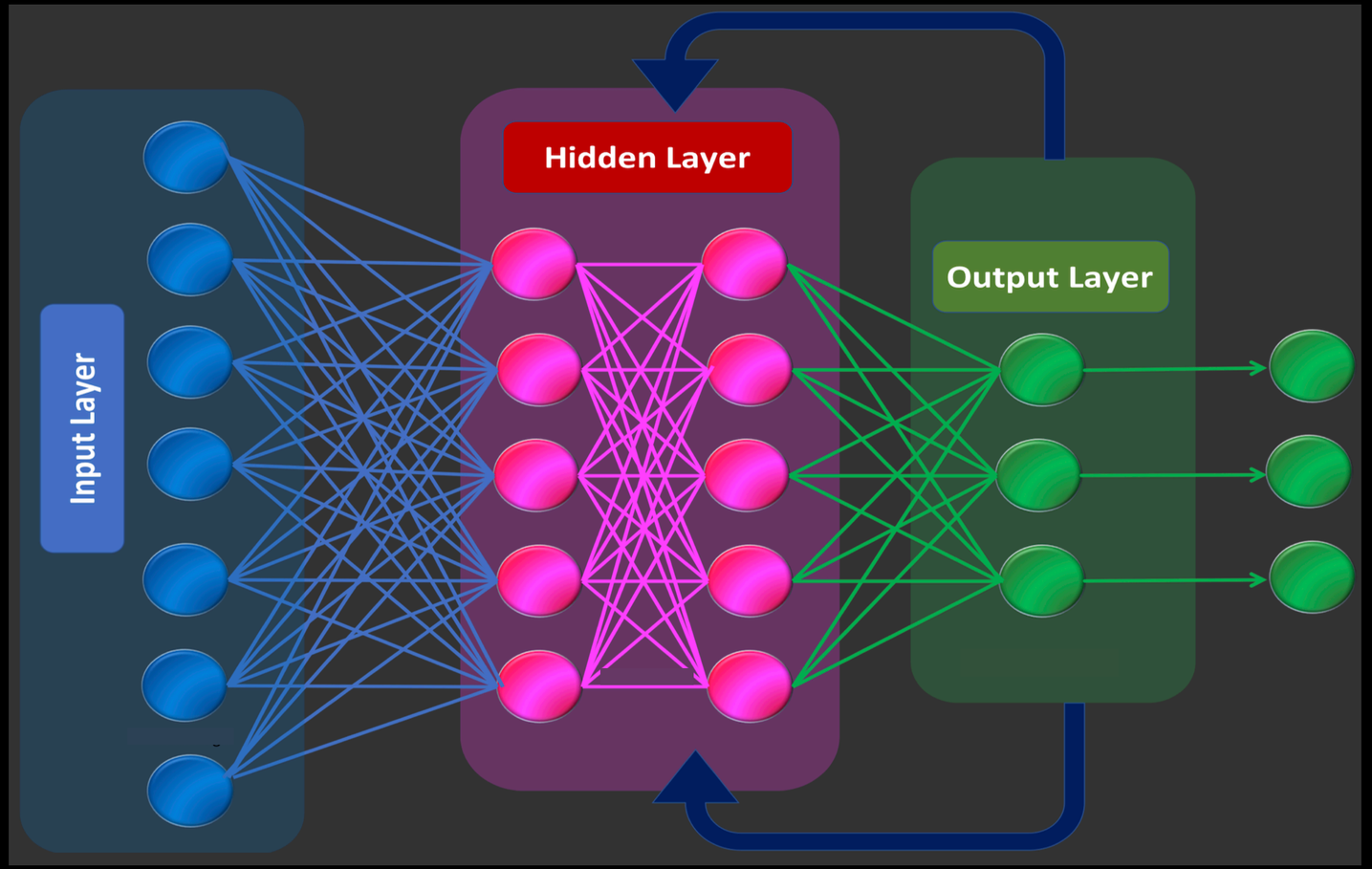
TENSORFLOW

https://www.tensorflow.org

Rafael Vieira Coelho

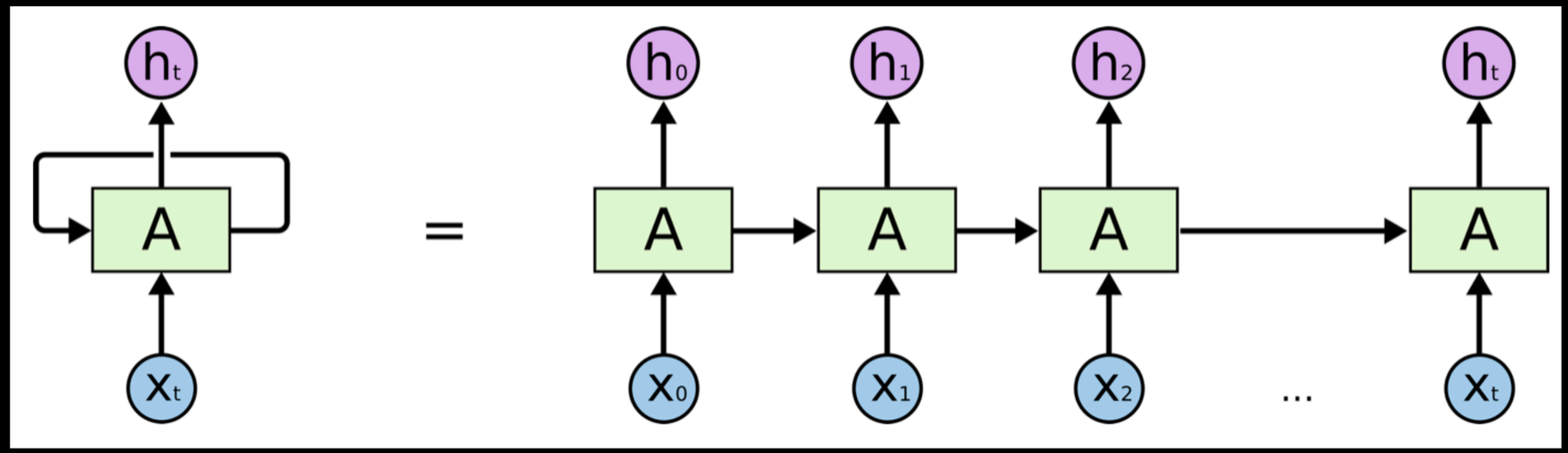
TENSORFLOW

REDES RECORRENTES



A) ARQUITETURA DE REDES NEURAIS LONG SHORT TERM MEMORY (LSTM)

É um tipo de rede neural recorrente, que é usada em diversos cenários de Processamento de Linguagem Natural. Ela pode ser imaginada como múltiplas cópias da mesma rede, cada uma passando uma mensagem a um sucessor.



A LSTM é bem adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida.

O QUE É UMA REDE NEURAL LSTM?

- Os LSTMs contêm informações fora do fluxo normal da rede recorrente em uma célula fechada. As informações podem ser armazenadas, escritas ou lidas a partir de uma célula, como dados na memória de um computador.

- A célula toma decisões sobre o que armazenar, e quando permitir leituras, gravações e exclusões, através de portões abertos e fechados.

- Ao contrário do armazenamento digital em computadores, no entanto, esses portões são analógicos, implementados com a multiplicação de elementos por sigmóides, que estão todos na faixa de 0-1.

COMO FUNCIONAM OS PORTÕES (GATES) EM UMA LSTM?

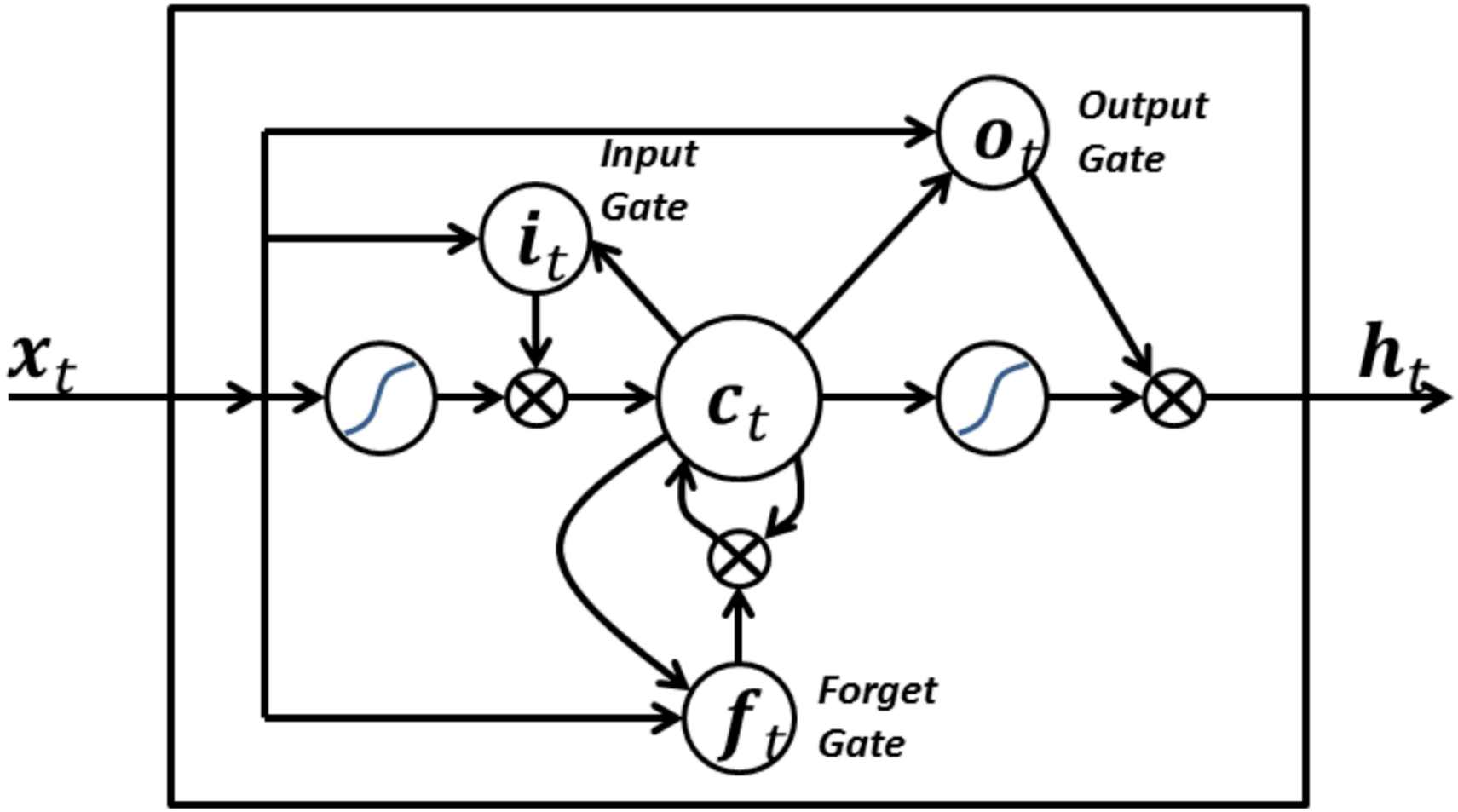
- Esses portões atuam sobre os sinais que recebem e, de forma semelhante aos nós da rede neural, eles bloqueiam ou transmitem informações com base em sua força e importação, que eles filtram com seus próprios conjuntos de pesos.

- Esses pesos, como os pesos que modulam a entrada e estados ocultos, são ajustados através do processo de aprendizagem das redes recorrentes.

- Ou seja, as células aprendem quando permitir que os dados entrem, saiam ou sejam excluídos através do processo iterativo de fazer suposições, calculando o erro durante o backpropagation e ajustando pesos através da descida do gradiente.

COMO FUNCIONAM OS PORTÕES (GATES) EM UMA LSTM?

O diagrama abaixo ilustra como os dados fluem através de uma célula de memória e são controlados por seus portões.



LSTM: CAMADA ÚNICA

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 64)          523840
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               66048
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 598,209
Trainable params: 598,209
Non-trainable params: 0
```

Long Short Term Memory (LSTM)

LSTM: DUAS CAMADAS

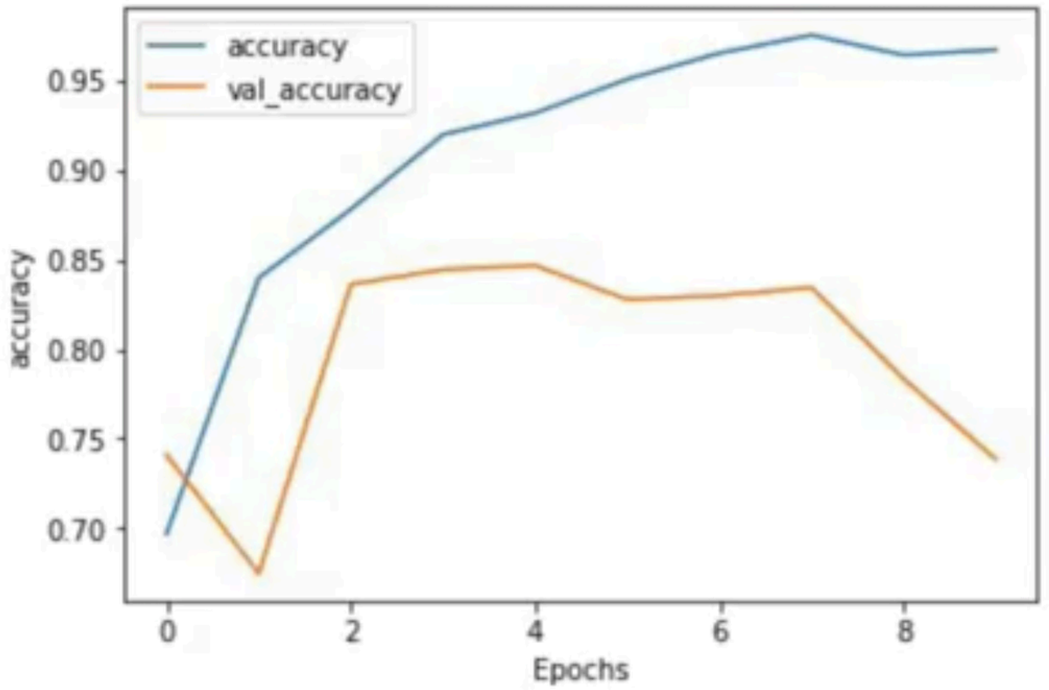
```python
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

É necessário estar com True a propriedade return_sequences para passar para a próxima camada bidirecional.

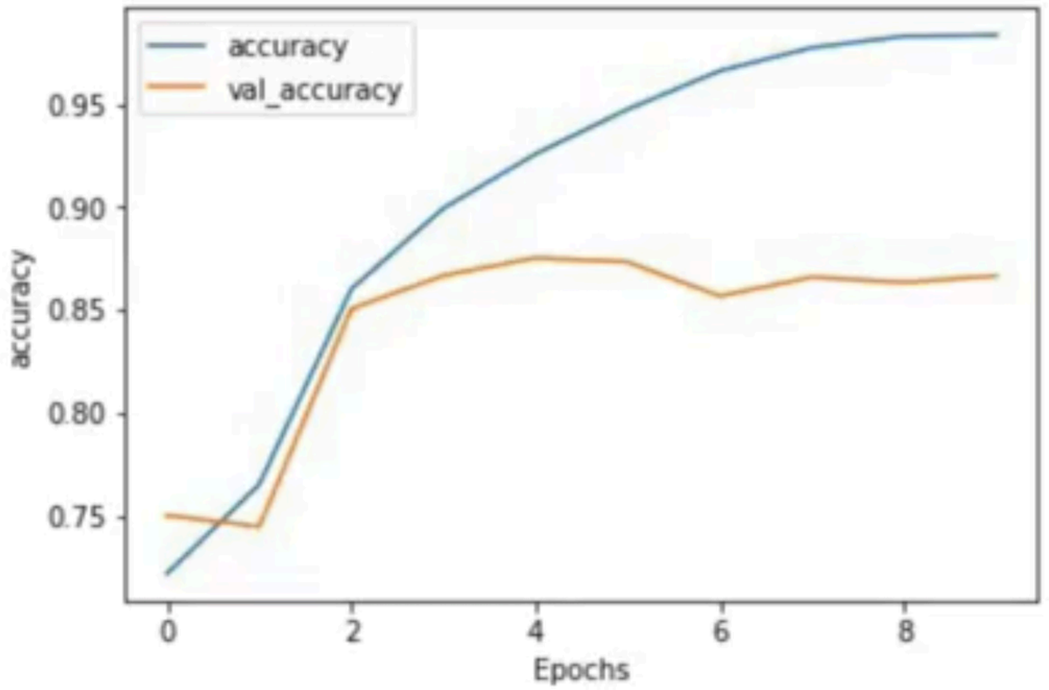
```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 64)          523840
_________________________________________________________________
bidirectional_2 (Bidirection (None, None, 128)         66048
_________________________________________________________________
bidirectional_3 (Bidirection (None, 64)                41216
_________________________________________________________________
dense_6 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 65
=================================================================
Total params: 635,329
Trainable params: 635,329
Non-trainable params: 0
```

LSTM: CAMADA ÚNICA X DUAS CAMADAS

10 Epochs : Accuracy Measurement



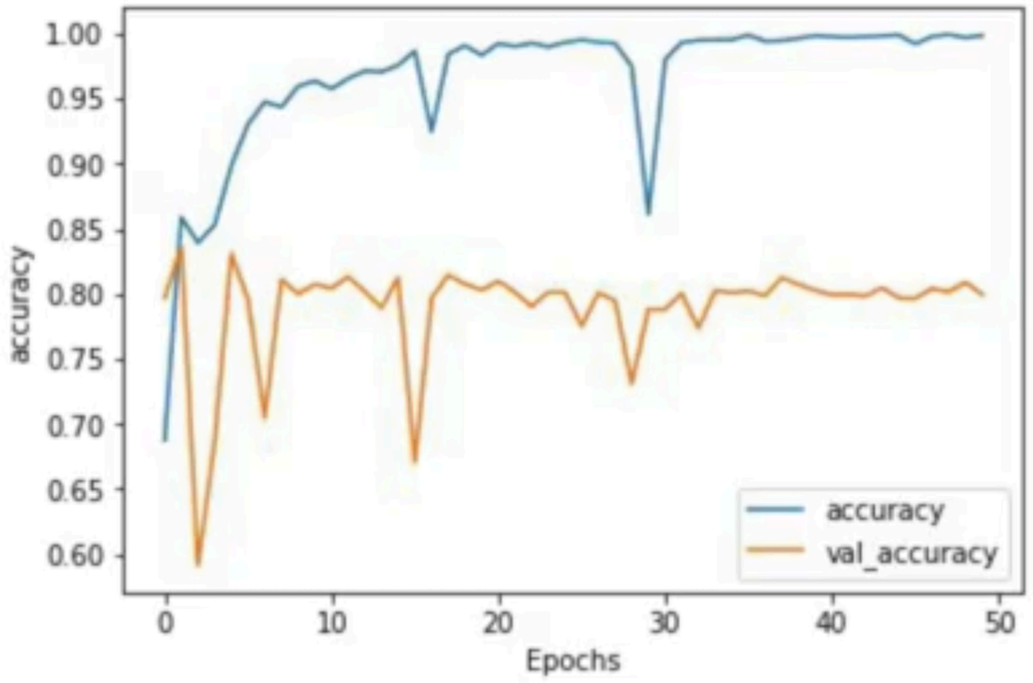
1 Layer LSTM



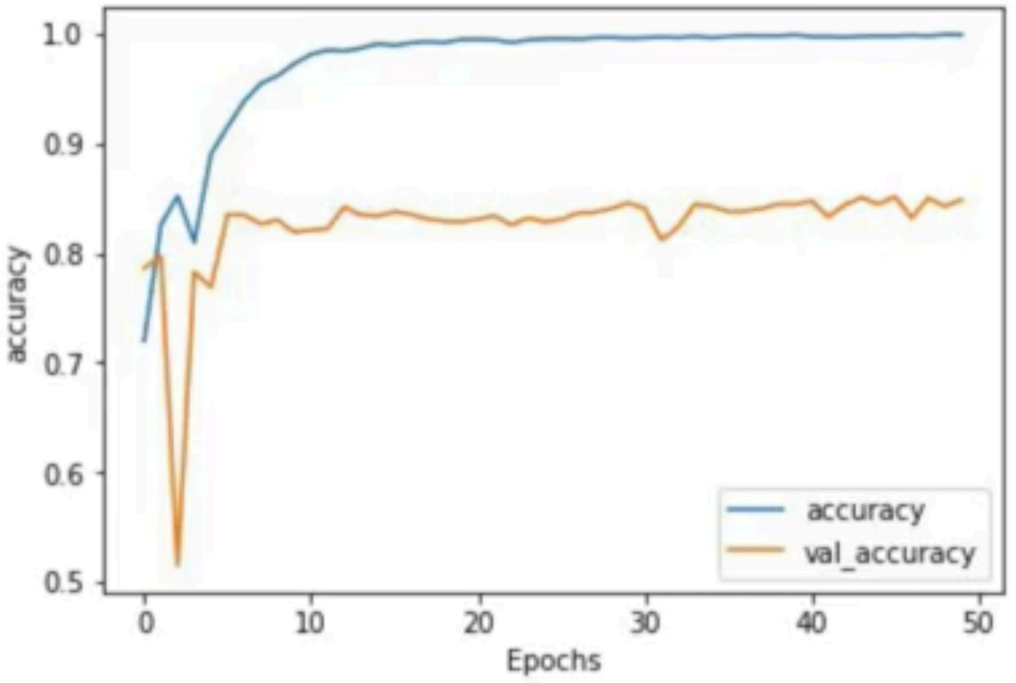
2 Layer LSTM

LSTM: CAMADA ÚNICA X DUAS CAMADAS

50 Epochs : Accuracy Measurement



1 Layer LSTM

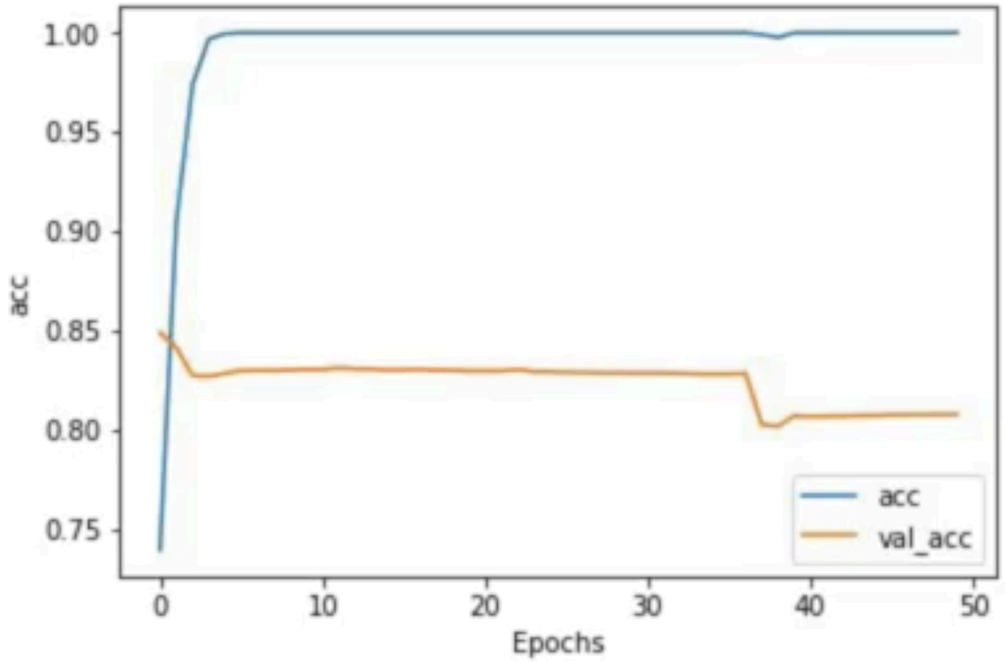


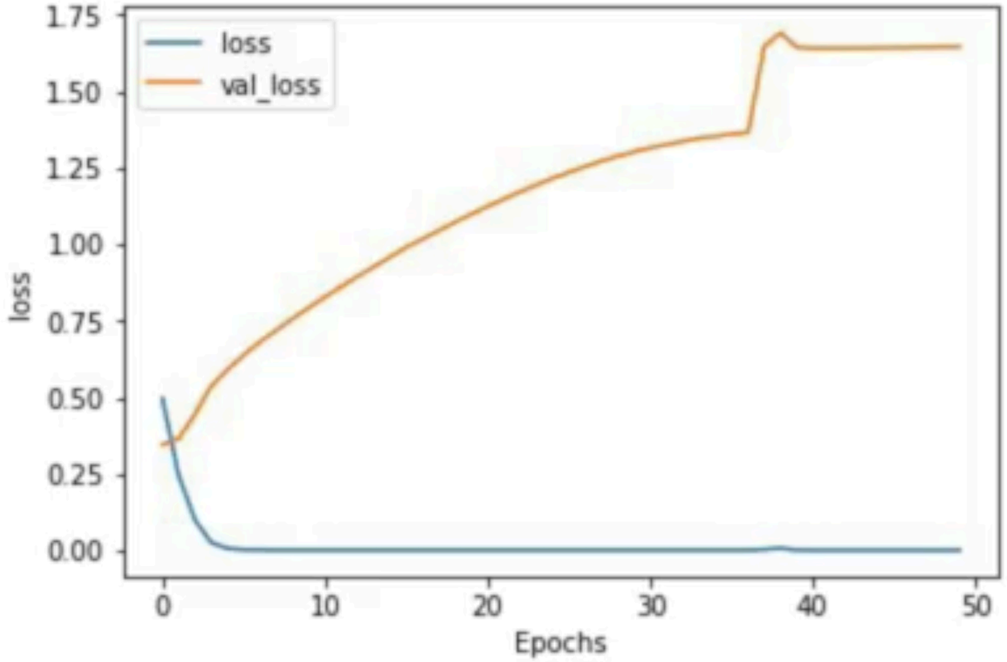
2 Layer LSTM

B) EXEMPLO USANDO CAMADA ACHATADA

```
imdb, info = tfds.load("imdb_reviews", with_info=True, as_supervised=True)



model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           160000
_________________________________________________________________
flatten (Flatten)            (None, 1920)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 11526
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 7
=================================================================
Total params: 171,533
Trainable params: 171,533
Non-trainable params: 0
```



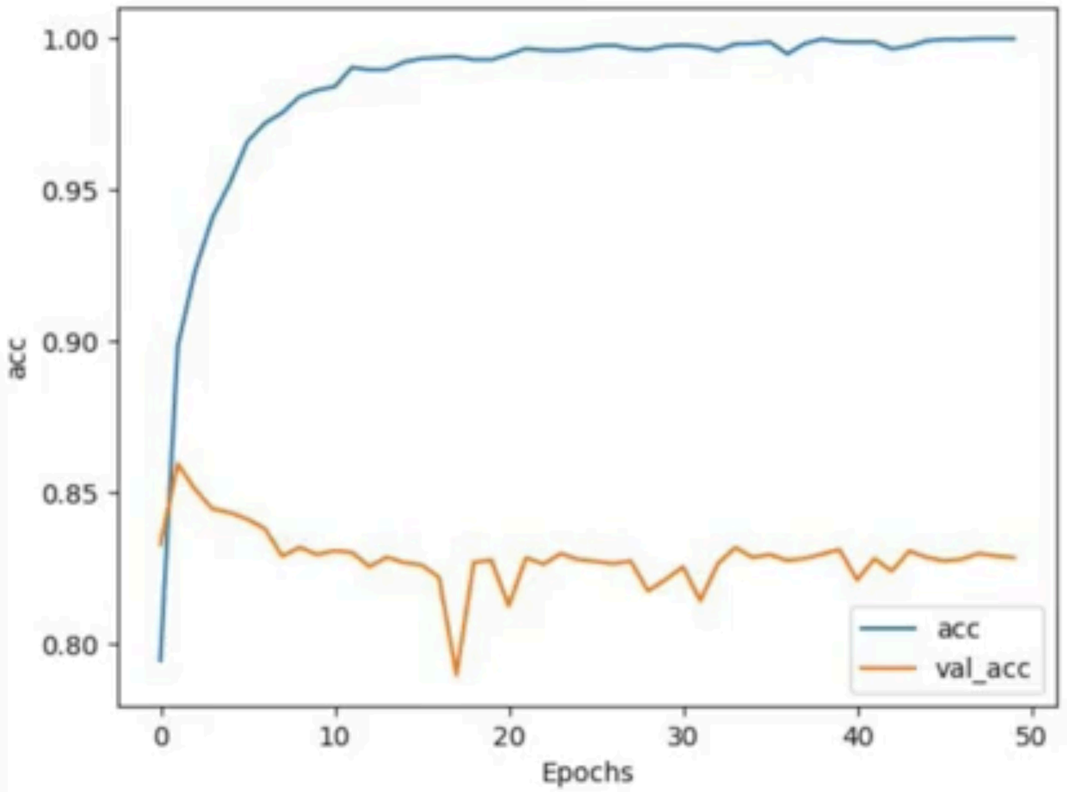


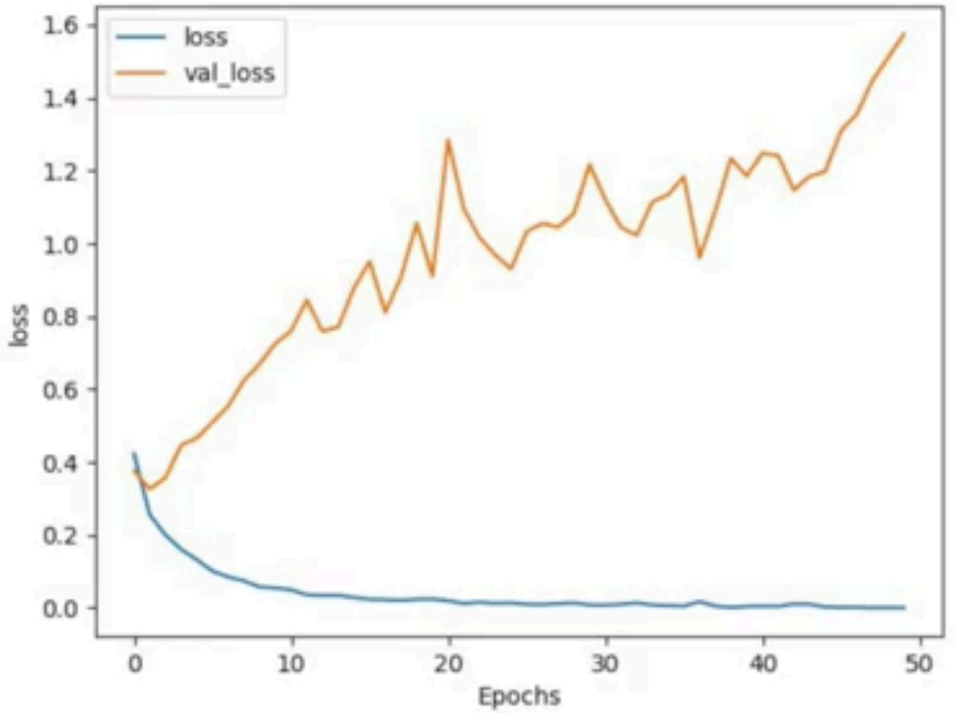
IMDB with Embedding-only : ~ 5s per epoch

Compartilhar

```
imdb, info = tfds.load("imdb_reviews", with_info=True, as_supervised=True)



# Model Definition with LSTM
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, 120, 16)           16000
_________________________________________________________________
bidirectional_7 (Bidirection (None, 64)                12544
_________________________________________________________________
dense_14 (Dense)             (None, 24)                1560
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 25
=================================================================
Total params: 30,129
Trainable params: 30,129
Non-trainable params: 0
```

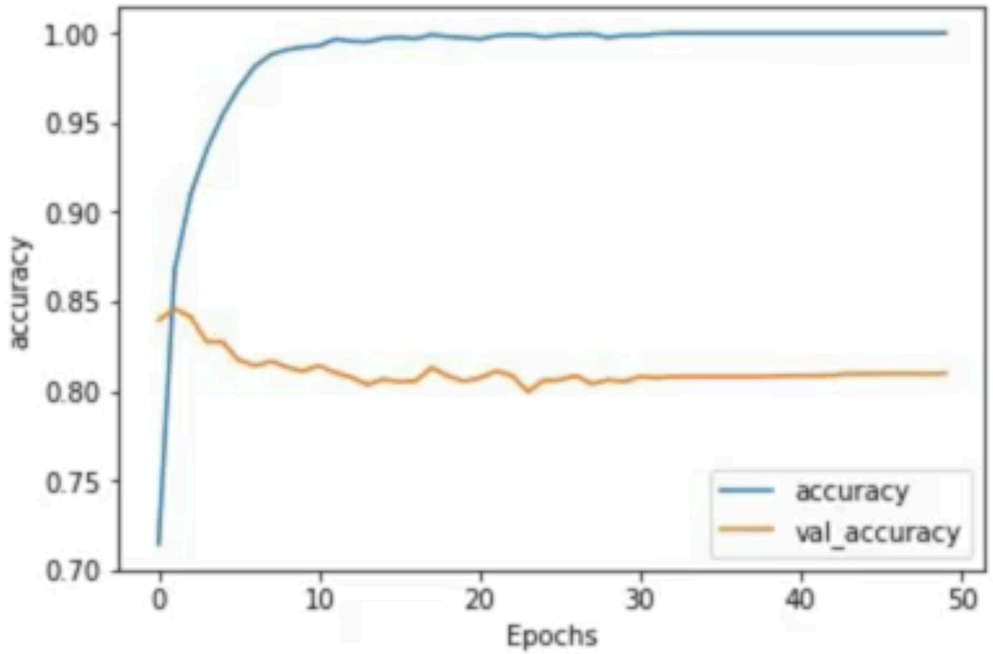


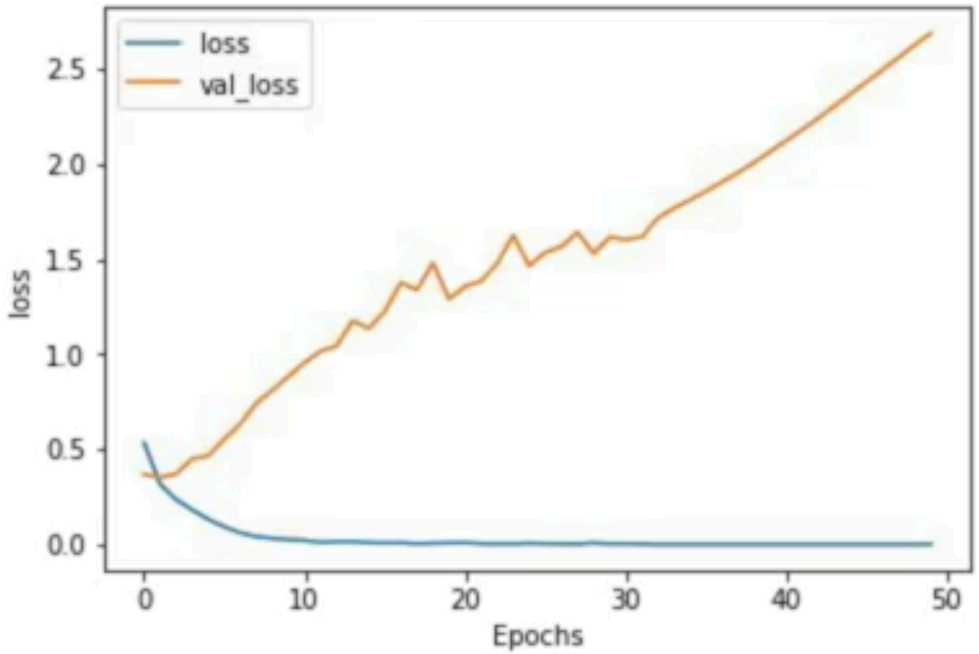


IMDB with LSTM ~43s per epoch

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32)),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 120, 16)           160000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                9600
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 7
=================================================================
Total params: 169,997
Trainable params: 169,997
Non-trainable params: 0
```

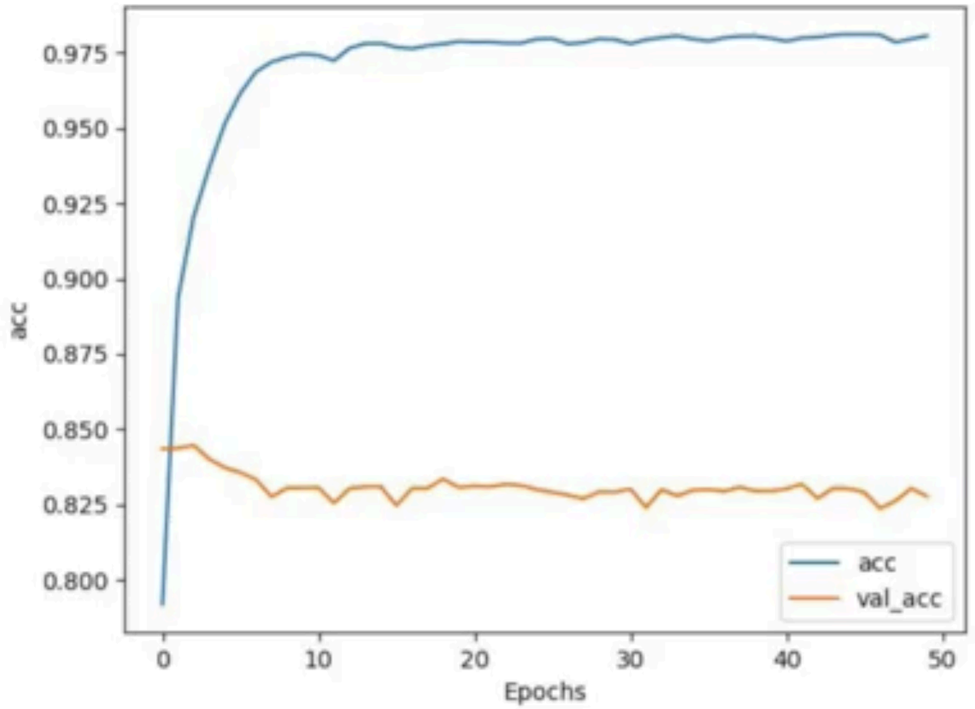


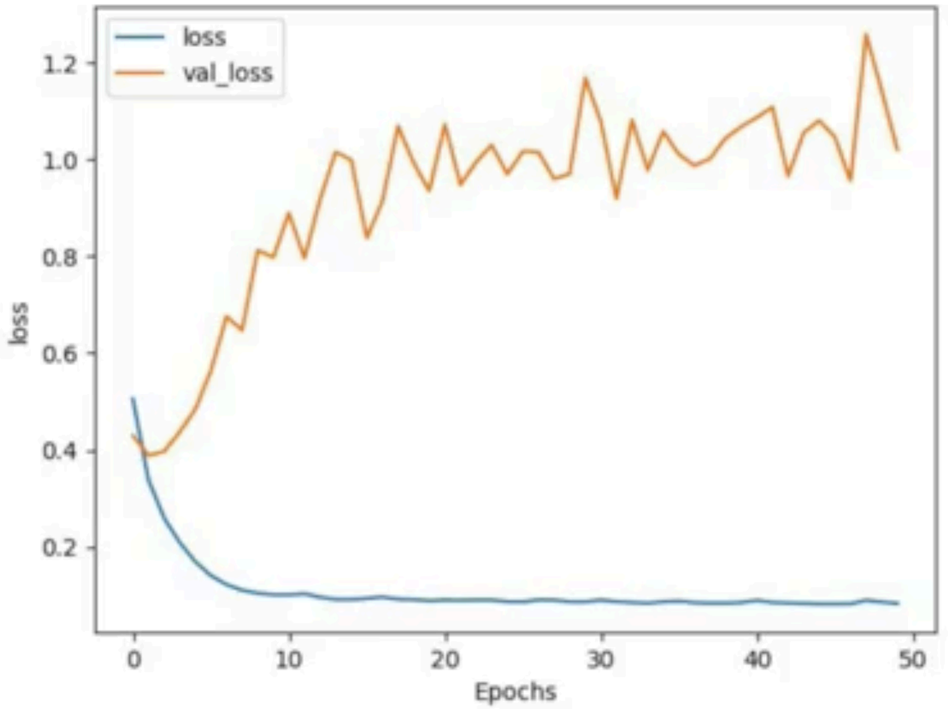


IMDB with GRU : ~ 20s per epoch

```
# Model Definition with Conv1D
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Conv1D(128, 5, activation='relu'),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           160000
_________________________________________________________________
conv1d (Conv1D)              (None, 116, 128)          10368
_________________________________________________________________
global_average_pooling1d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 6)                 774
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 7
=================================================================
Total params: 171,149
Trainable params: 171,149
Non-trainable params: 0
_________________________________________________________________
```





IMDB with CNN : ~ 6s per epoch

LINKS ÚTEIS

https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%203%20-%20Lesson%201a.ipynb

https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%203%20-%20Lesson%201b.ipynb

https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%203%20-%20Lesson%202d.ipynb#scrollTo=nHGYuU4jPYaj

EXERCÍCIOS

Explore os modelos criados abaixo:

https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%203%20-%20Lesson%202.ipynb#scrollTo=g9DC6dmLF8DC

https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%203%20-%20Lesson%202c.ipynb#scrollTo=g9DC6dmLF8DC

DÚVIDAS?

RAFAELVC2@GMAIL.COM