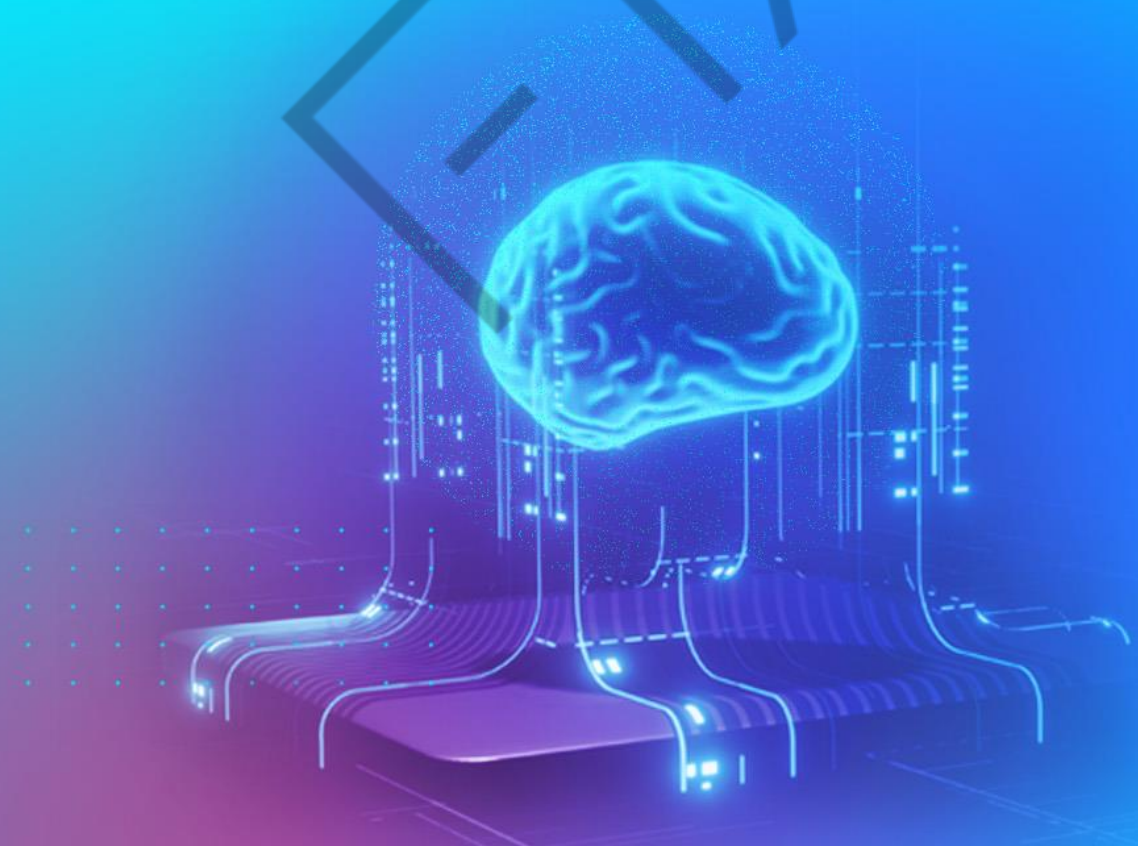
PRESTO! DATA VIZ!

LINGUAGEM NATURAL E ANÁLISE DE SENTIMENTOS



2

LISTA DE FIGURAS

Figura 1 – IA, ML,DL 6
Figura 2 – Perceptron.....7
Figura 3 – MLP – TensorFlow 7
Figura 4 – Pesquisa Google.....9
Figura 5 – RNN versus ANN 10
Figura 6– Desenrolando RNN 10
Figura 7– LSTM.....11
Figura 8 – GRU 12

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – *Review* filmes.....14
Código-fonte 2 – Composição musical.....16

SUMÁRIO

1 LINGUAGEM NATURAL E ANÁLISE DE SENTIMENTOS 5
1.1 Deep Learning 5
1.2 RNN – Redes Neurais Recorrentes 8
1.3 Diferença entre Rede Neural Recorrente e Rede Neural Artificial 9
1.4 Arquiteturas de redes RNN – LSTM e GRU 11
1.5 Aplicação de RNN – LSTM e GRU 12
REFERÊNCIAS 17

1 LINGUAGEM NATURAL E ANÁLISE DE SENTIMENTOS

A utilização de modelos de aprendizado profundo (Deep Learning) para problemas de processamento de linguagem natural teve grande avanço nos últimos anos. Neste capítulo, vamos entrar no mundo do Deep Learning para processamento de texto por meio da utilização de algoritmos de redes neurais recorrentes RNN. O processamento de texto tem aplicações em diversas problemáticas da vida real, entre eles:

- Classificação de documentos: análises de grandes volumes de processos jurídicos para classificação de parecer e sentença.
- Comparação de documentos: avaliação de textos para detecção de cópias e plágios.
- Tradutor de idioma: identificação de contextos no encadeamento de palavras para traduzir de um idioma para outro.
- Análises de sentimentos, ou classificação de sentimento em comentários de redes sociais.
- Criação de textos, criação de novos textos com base no padrão de escrita de um autor usado para criação de matérias de revistas e editoriais.

Vamos entender, neste capítulo, um pouco dos conceitos e algoritmos para tornar isso realidade.

1.1 Deep Learning

O aprendizado profundo, ou Deep Learning, é uma subárea do aprendizado de máquina ML baseado em algoritmos que utilizam redes neurais artificiais (RNA), como demonstra o diagrama de Venn (Figura “IA, ML, DL”). O conceito de RNA surge por volta da década de 1940 e o primeiro modelo, batizado de Perceptron, em 1959, foi capaz de modelar matematicamente, inspirado na sinapse de um neurônio humano.

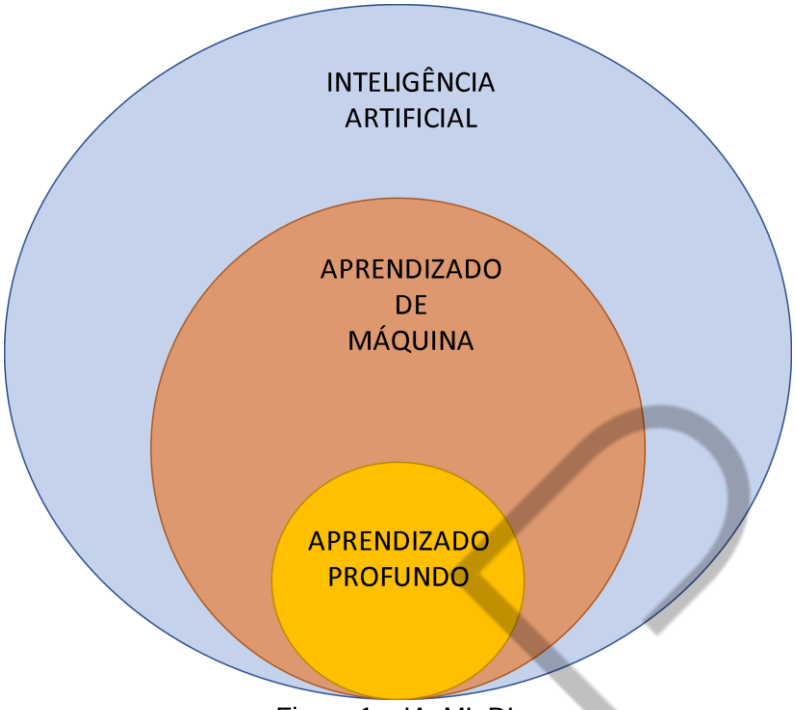


Figura 1 – IA, ML,DL
Fonte: Elaborada pelo autor (2022)

O Perceptron pode ser dividido em três etapas e o processamento segue os seguintes passos:

1. O modelo recebe um conjunto de sinais (X) como entrada.
2. Cada sinal é multiplicado por um número que pondera a influência desse sinal na saída da unidade. Chamaremos o conjunto desses números de **pesos** (W).
3. É realizada uma soma ponderada de todos os sinais, produzindo um número (valor escalar).
4. Se o resultado for maior que um limiar, a unidade produz um valor resultante determinado.

Esses passos podem ser representados pela Figura “Perceptron” a seguir:

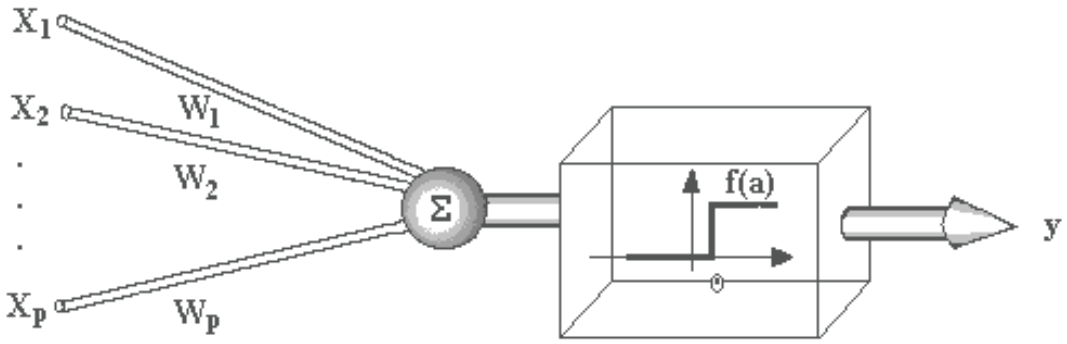


Figura 2 – Perceptron
Fonte: Estudos Avançados (2021)

Uma rede neural é composta por diversos neurônios que são associados em camadas, dando o nome de Rede Multi Layer Perceptron ou MLP. Essa associação é feita quando neurônios recebem os dados de entrada, processam e propagam sua saída para outro conjunto de neurônios e assim por diante. Os conjuntos de neurônios são conhecidos como camadas (Layers); uma rede MLP possui uma camada de entrada e uma camada de saída e entre essas camadas existem as camadas ocultas (Hidden Layers), que podem ser uma ou mais camadas associadas. Todos os neurônios de uma camada oculta servem de entrada para a camada seguinte, por isso podemos chamar uma rede MLP de rede densa ou totalmente conectada também. Como podemos ver na Figura “MLP – TensorFlow”, para o classificador com seis camadas ocultas e oito neurônios em cada camada oculta .

Figura 3 – MLP – TensorFlow
Fonte: Elaborada pelo autor (2022)

Para saber mais: https://playground.tensorflow.org

Agora você deve estar pensando: mas e o *"deep"* de Deep Learning, de onde vem? O *deep* está relacionado ao número de camadas ocultas que existem na rede neural; quanto mais camada, mais deep é a rede neural. Redes neurais modernas possuem facilmente mais de dez camadas ocultas. No nosso caso, Deep Learning é um framework (TensorFlow) utilizado para treinamento e representação de aprendizado por meio de dados (textos).

O uso de técnicas de Deep Learning é muito vasto e pode ser usado em vários problemas da ciência de dados, incluindo Classificação, Regressão, Aprendizado por Reforço, Redução de Dimensionalidade, Clusterização e Análise de Sentimentos.

1.2 RNN – Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNNs) são uma técnica de Deep Learning muito empregada em aplicações de Processamento de Linguagem Natural (NLP) em que os dados são processados sequencialmente, resultando em uma análise aprimorada de conjuntos de dados complexos, por meio da modelagem em séries temporais dos dados de entrada. De certa forma, podemos compreender que uma RNNs é inspirada na memória humana, pois permite o entendimento do relacionamento entre os dados do passado com os atuais, como fazemos. Para compreender melhor, vamos ver um exemplo: tente analisar a frase a seguir retida do livro *O guia dos Mochileiros das Galáxias*:

"42. mais e o vida, sobre grande resposta é tudo universo a questão a a"

Não faz sentido nenhum, vamos ordenar a sequência das palavras e analisar novamente:

"A resposta à Grande Questão sobre a Vida, o Universo e Tudo Mais é 42."

Agora sim! Conseguimos compreender a frase, extraindo informação dos dados, isso porque, em alguns casos, a extração da informação não parte apenas dos dados, mas também da forma que é apresentada. A informação é entendida por uma sequência de palavras, ou seja, não descartamos as palavras anteriores ao ler

novas palavras, nós compreendemos a nova palavra a partir das anteriores, essa é a característica de memória.

A utilização de RNN é largamente utilizada atualmente nas mais diversas aplicações, por exemplo, quem nunca usou o autocompletar do Google (Figura “Pesquisa Google”).

Figura 4 – Pesquisa Google
Fonte: Elaborada pelo autor (2022)

1.3 Diferença entre Rede Neural Recorrente e Rede Neural Artificial

A diferença principal entre uma Rede Neural Recorrente para uma Rede Neural Artificial é que a saída de uma RNN retroalimenta a entrada do próprio neurônio, como pode ser visto na Figura “RNN versus ANN”, em que, à direita, temos uma rede neural artificial e, à esquerda, uma RNN com a retroalimentação.

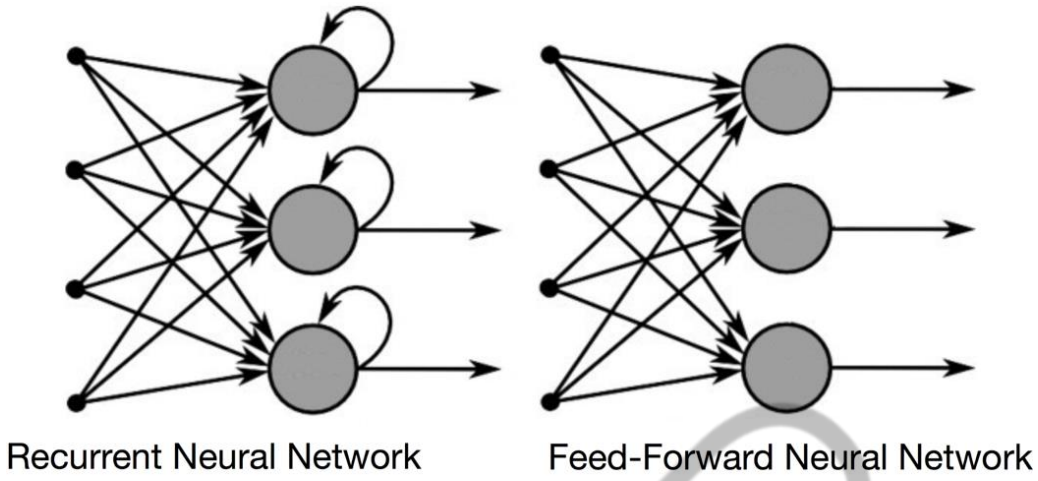


Figura 5 – RNN versus ANN
Fonte: Radhakrishnan (2017)

A saída atual do neurônio ocorre por meio da função de ativação das entradas atuais multiplicadas pelos seus pesos somados à saída anterior multiplicada pelo seu peso, como demonstra a equação abaixo.

$$h_t = \emptyset(Wx_t + Uh_{t-1}) \quad (1)$$

Para facilitar a interpretação da equação acima, vamos pegar apenas um neurônio e iterar os estados ao longo do tempo t, como vemos na Figura "Desenrolando RNN": à esquerda, temos nosso neurônio e X representa o vetor de entrada, U o vetor de peso, V o vetor de saída e W o efeito memória da retroalimentação ao longo do tempo. À direta conseguimos visualizar ao longo do tempo t-1,t, t+1 a evolução; note que o estado de saída futuro depende da saída atual, assim como a saída atual depende da saída no estado anterior de tempo.

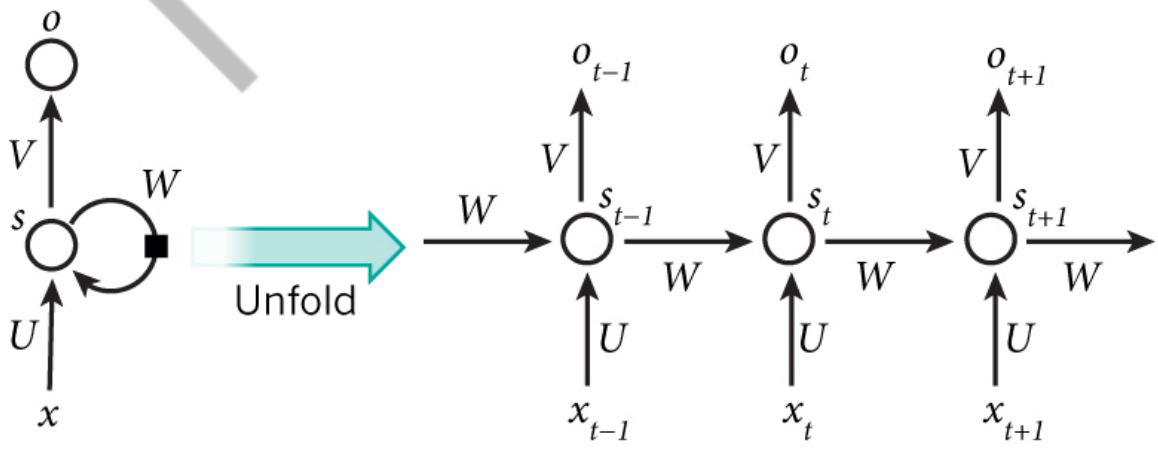


Figura 6– Desenrolando RNN
Fonte: Abdelrahman Ayad (2018)

1.4 Arquiteturas de redes RNN – LSTM e GRU

Na prática, uma RNN não possui capacidade de capturar informações de muitas etapas anteriores (memória de longo prazo), devido à grande complexidade desse processamento, um problema chamado de explosão de gradiente; por essa razão, as redes LSTM são muito utilizadas.

A rede LSTM, ou Long Short Term Memory, é um tipo de rede recorrente muito utilizada em tarefas de processamento de linguagem natural. Essa rede é útil para classificar e prever séries temporais com intervalos de tempo e duração desconhecidos. Basicamente, trata-se de uma RNN que computa os estados ocultos de forma um pouco diferente. Possui uma estrutura em cadeia que contém quatro redes neurais e diferentes blocos de memória chamados células, como pode ser visto na Figura “LSTM”.

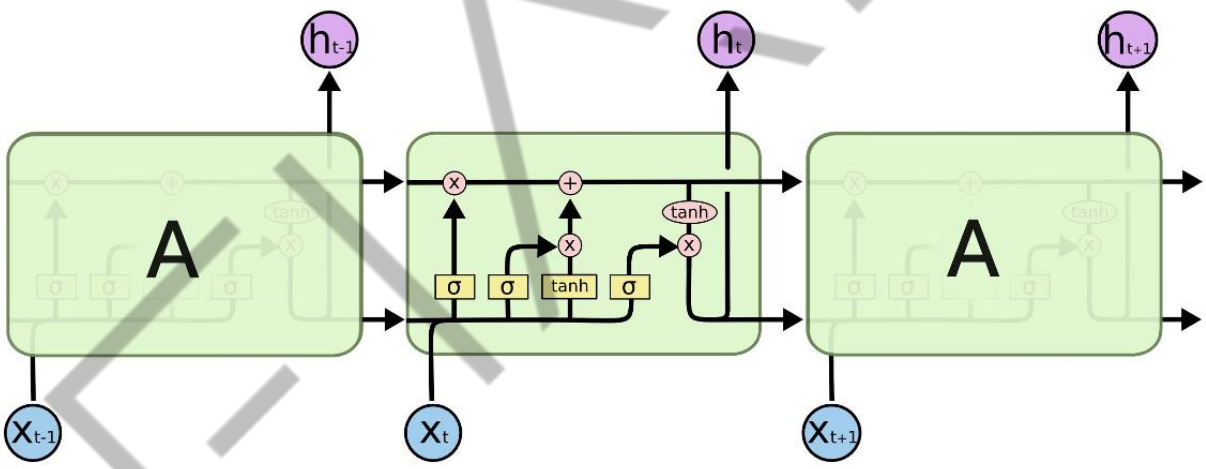


Figura 7– LSTM
Fonte: Fu (2017)

A arquitetura GRU é uma solução para memória de curto prazo e possui portões para regular o fluxo de informação. Porém, uma diferença é a quantidade de gates comparada com LSTM, como pode ser visto na Figura “GRU”.

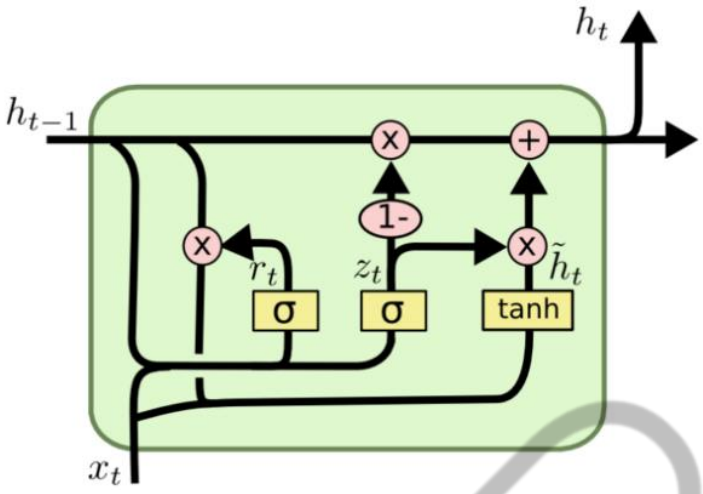


Figura 8 – GRU
Fonte: Fu (2017)

1.5 Aplicação de RNN – LSTM e GRU

Aplicaremos os conceitos de LSTM e GRU para treinar um conjunto de dados de comentários de filmes da plataforma IMDB para classificar se a *review* foi positiva ou negativa sobre o filme. A predição desse modelo é entendida como positiva se os valores de saída forem maiores que 0 e negativa caso contrário. Assim podemos facilmente predizer novas *reviews*, classificando como positiva ou negativa automaticamente.

```
import numpy as np

import tensorflow_datasets as tfds
import tensorflow as tf

BUFFER_SIZE = 10000
BATCH_SIZE = 64
VOCAB_SIZE=1000

# Baixando e carregando o conjunto de dados
dataset, info = tfds.load('imdb_reviews', with_info=True, as_supervise
d=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

```
# Dividindo e embaralhando os dados
train_dataset = train_dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE).p
refetch(tf.data.AUTOTUNE)
test_dataset = test_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUN
E)

# Realizando o pre-processamento
encoder = tf.keras.layers.experimental.preprocessing.TextVectorization
(
    max_tokens=VOCAB_SIZE)
encoder.adapt(train_dataset.map(lambda text, label: text))
# Criando o modelo
model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(
        input_dim=len(encoder.get_vocabulary()),
        output_dim=64, mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])

# Compilando o model
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True
),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])

# Treinando o modelo
history = model.fit(train_dataset, epochs=10,
                    validation_data=test_dataset,
                    validation_steps=30)

# Validando o modelo
test_loss, test_acc = model.evaluate(test_dataset)
print('Test Loss: {}'.format(test_loss))
print('Test Accuracy: {}'.format(test_acc))
```

```
# Realizando uma predição
sample_text = ('The movie was cool. The animation and the graphics '
               'were out of this world. I would recommend this movie.'
)
predictions = model.predict(np.array([sample_text]))
print(predictions)
```

Código-fonte 1 – *Review* filmes
Fonte: Elaborado pelo autor (2022)

As RNNs como LSTM podem ser utilizadas para produzir textos, em que cada letra ou palavra gerada se utiliza da memória de todas as outras geradas até então. Como sugestão, crie um corpus com algumas letras de músicas do seu artista predileto.

```
import tensorflow as tf

from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectio
nal
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

import numpy as np
import matplotlib.pyplot as plt


# Carrega a base de dados
data = open('./musicas.txt').read()
corpus = data.lower().split("\n")
tokenizer = Tokenizer()

tokenizer.fit_on_texts(corpus)
print(tokenizer.word_index)

total_words = len(tokenizer.word_index) + 1
print("Total de palavras: ",total_words)

# tokenize o texto, transforma cada palavra em uma lista de numeros in
```

```
teiros

input_sequences = []

for line in corpus:
    token_list = tokenizer.texts_to_sequences([line])[0]
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i + 1]
        input_sequences.append(n_gram_sequence)

print(input_sequences)

# pad sequences
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences,
                                         maxlen=max_sequence_len,
                                         padding='pre'))
print(input_sequences)

# cria predictors and label
xs, labels = input_sequences[:, :-1], input_sequences[:, -1]
ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)

# Cria a rede
model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len -
 1))
model.add(Bidirectional(LSTM(150)))
model.add(Dense(total_words, activation='softmax'))


model.summary()

adam = Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=['accuracy'])

history = model.fit(xs, ys, epochs=100, verbose=1)

plt.plot(history.history['accuracy'])
plt.xlabel("Epochs")
plt.ylabel('accuracy')
plt.show()

#Teste

seed_text = "quem sabe eu"
```

```
next_words = 100

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list],
                               maxlen=max_sequence_len - 1,
                               padding='pre')
    predicted = np.argmax(model.predict(token_list), axis=-1)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)
```

Código-fonte 2 – Composição musical
Fonte: Elaborado pelo autor (2022)

REFERÊNCIAS

AMIDI, S. **CS 229:** aprendizado de máquina. [S.d.]. Disponível em: https://stanford.edu/~shervine/l/pt/teaching/cs-229/dicas-truques-aprendizado-maquina. Acesso em: 9 jan. 2022.

BENINI, L. A **Recurrent Neural Network Speech Recognition Chip.** 2020. Disponível em: https://iis-projects.ee.ethz.ch/index.php/A_Recurrent_Neural_Network_Speech_Recognition_Chip. Acesso em: 9 maio 2022.

DUDA, R.; HART, P.; STORK, D. **Pattern Classification.** 2. ed. [S.l.]: Wiley, 2001.

ESTUDOS AVANÇADOS. São Paulo: Instituto de Estudos Avançados da Universidade de São Paulo, v. 35, n. 101, jan./abr. 2021.

FU, J. *et al.* Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. **IEEE Access,** v. 7, p. 57.078-57.087, 2019. DOI: 10.1109/ACCESS.2019.2912621.

LUGER, G. F. **Inteligência artificial.** 6. ed. São Paulo: Pearson Education do Brasil, 2013. (Biblioteca virtual.)

MITCHELL, T. M. **Machine Learning.** 1. ed. Nova York, United States: McGraw-Hill Education, 1997.

RADHAKRISHNAN, P. Introduction to recurrent neural networks. 2017. **Towards Data Science.** Disponível em: https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3. Acesso em: 9 maio 2022.

RUSSEL, S.; NORVIG, P. **Inteligência artificial.** 3. ed. Rio de Janeiro: Campus, 2012.