Ejemplo RNN para Predecir la siguiente palabra

Importar Librerias

```
from keras.layers import Dense, Activation
from keras.layers.recurrent import SimpleRNN
from keras.models import Sequential
import numpy as np
import tensorflow as tf
```

Hiperparametros

```
hidden_neurons = 50
my_optimizer ="sgd"
batch_size = 60
error_function = "mean_squared_error"
output_nonlinearity = "softmax"
cycles = 5
epochs_per_cycle = 3
context = 3
```

La variable hidden_neurons indica cuántas unidades ocultas vamos a utilizar. La variable optimizer define qué optimizador de Keras vamos a utilizar, y en este caso es el descenso de gradiente estocástico La variable batch_size define el tamaño del lote simplemente dice cuántos ejemplos usaremos para una sola iteración del descenso del gradiente estocástico. La variable rror_function = "mean_squared_error" le dice a Keras que use el MSE La función de activación output_nonlinearity, es la función de activación softmax o no linealidad, con su nombre Keras "softmax".Básicamente transforma un vector z con arbitraria valores reales a un vector con valores que van de 0 a 1, y son tales que todos suman 1.

Data

```
def create_tesla_text_from_file(textfile="Hawking.txt"):
  clean_text_chunks = []
  with open(textfile, 'r', encoding='utf-8') as text:
    for line in text:
        clean_text_chunks.append(line)
  clean_text = ("".join(clean_text_chunks)).lower()
  text_as_list = clean_text.split()
```

```
  return text_as_list
text_as_list = create_tesla_text_from_file()
```

Nota: se hace uso de este dataset dado que no se encontro el del Ejemplo del libro

La funcion create_tesla_text_from_file(textfile= "tesla.txt") Abre y se lee el archivo, este lo devuelve línea por línea, por lo que Se guarda estas líneas en la variable clean_text_chunks. Luego se pegan todas estas juntas en una cadena grande llamada clean_text, y luego se cortan
en palabras individuales y esto es lo que hace dicha funcion toda la función Y el resultado de todo ese proceso se guarda en text_as_list Ahora, tenemos todo nuestro texto en una lista, donde cada elemento individual es una palabra.

Procesar Data

Tenga en cuenta que puede haber repeticiones de palabras aquí, y eso está perfectamente bien, ya que esto será manejado por la siguiente parte del código:

```
distinct_words = set(text_as_list)
number_of_words = len(distinct_words)
word2index = dict((w, i) for i, w in enumerate(distinct_words))
index2word = dict((i, w) for i, w in enumerate(distinct_words))
```

La variable number_of_words cuenta el número de palabras en el texto. La variable word2index crea un diccionario con palabras únicas como claves y su posición en el texto como valores. La variable index2word hace exactamente lo contrario, crea un diccionario donde las posiciones son claves y las palabras son valores.

Ahora se crea una función que crea una lista de palabras de entrada y una lista de etiquetas de palabras del texto original, que debe tener la forma de una lista de palabras individuales. Lo que buscamos es hacer un estructura 'entrada'/'etiqueta' para predecir la siguiente palabra, y lo hacemos descomponiendo esta oración en una matriz. Esta funcion toma un texto en forma de lista, crea la lista de palabras de entrada y la lista de palabras de etiquetas y devuelve los dos.

```
def create_word_indices_for_text(text_as_list):
  input_words = []
  label_word = []
  for i in range(0,len(text_as_list) - context):
    input_words.append((text_as_list[i:i+context]))
    label_word.append((text_as_list[i+context]))
  return input_words, label_word
input_words,label_word = create_word_indices_for_text(text_as_list)

input_vectors = np.zeros((len(input_words), context, number_of_words), dtype=np.int16)
vectorized_labels = np.zeros((len(input_words), number_of_words), dtype=np.int16)
```

Este código produce tensores 'en blanco', poblados por ceros.

El tensor input_vectors es técnicamente una 'matriz' con tres dimensiones, El vectorized_labels es el mismo, solo aquí no tenemos tres o n palabras especificadas por la variable context, sino solo una sola, la palabra etiqueta, por lo que necesitamos una dimensión menos en el tensor.

Ahora bien necesitamos poner ambos tensores en el lugar apropiado, para esto se hace lo siguiente, donde se busca poiner en 1s los tensores

```
for i, input_w in enumerate(input_words):
  for j, w in enumerate(input_w):
    input_vectors[i, j, word2index[w]] = 1
    vectorized_labels[i, word2index[label_word[i]]] = 1
```

Modelo Definimos el modelo RNN

```
model = Sequential()
model.add(SimpleRNN(hidden_neurons, return_sequences=False,
input_shape=(context,number_of_words), unroll=True))
model.add(Dense(number_of_words))
model.add(Activation(output_nonlinearity))
model.compile(loss=error_function, optimizer=my_optimizer)
```

Todos los parametro que utiliza se definieron antes y se explicaron

Entrenar y Probando el Modelo

Se entrena y se prueba el modelo

```
for cycle in range(cycles):
  print("> - <" * 50)
  print("Cycle: %d" % (cycle+1))
  model.fit(input_vectors, vectorized_labels, batch_size = batch_size,
epochs = epochs_per_cycle)
  test_index = np.random.randint(len(input_words))
  test_words = input_words[test_index]
  print("Generating test from test index %s with words %s:" % (test_index,
test_words))
  input_for_test = np.zeros((1, context, number_of_words))
  for i, w in enumerate(test_words):
    input_for_test[0, i, word2index[w]] = 1
  predictions_all_matrix = model.predict(input_for_test, verbose = 0)[0]
  predicted_word = index2word[np.argmax(predictions_all_matrix)]
  print("THE COMPLETE RESULTING SENTENCE IS: %s %s" % ("".join(test_words),
predicted_word))
  print()
```

```
> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <>
```

```
Cycle: 1
Epoch 1/3
1141/1141 [==============================] - 50s 44ms/step - loss: 1.0297e-04
Epoch 2/3
1141/1141 [==============================] - 47s 41ms/step - loss: 1.0297e-04
Epoch 3/3
1141/1141 [==============================] - 49s 43ms/step - loss: 1.0297e-04
Generating test from test index 59217 with words ['que', 'sea', 'posible']:
THE COMPLETE RESULTING SENTENCE IS: queseaposible desmoronarían.

> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <>
Cycle: 2
Epoch 1/3
1141/1141 [==============================] - 47s 42ms/step - loss: 1.0297e-04
Epoch 2/3
1141/1141 [==============================] - 49s 43ms/step - loss: 1.0297e-04
Epoch 3/3
1141/1141 [==============================] - 48s 42ms/step - loss: 1.0297e-04
Generating test from test index 57175 with words ['velocidades', 'diferentes.', 'esto']
THE COMPLETE RESULTING SENTENCE IS: velocidadesdiferentes.esto dicen

> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <>
Cycle: 3
Epoch 1/3
1141/1141 [==============================] - 50s 44ms/step - loss: 1.0297e-04
Epoch 2/3
1141/1141 [==============================] - 48s 42ms/step - loss: 1.0297e-04
Epoch 3/3
1141/1141 [==============================] - 49s 43ms/step - loss: 1.0297e-04
Generating test from test index 53119 with words ['formen', 'ningún', 'tipo']:
THE COMPLETE RESULTING SENTENCE IS: formenningúntipo actualidad

> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <>
Cycle: 4
Epoch 1/3
1141/1141 [==============================] - 50s 44ms/step - loss: 1.0297e-04
Epoch 2/3
1141/1141 [==============================] - 49s 43ms/step - loss: 1.0297e-04
Epoch 3/3
1141/1141 [==============================] - 49s 43ms/step - loss: 1.0297e-04
Generating test from test index 30295 with words ['las', 'estrellas', 'masivas.']:
THE COMPLETE RESULTING SENTENCE IS: lasestrellasmasivas. satisfactorio,

> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <> - <>
Cycle: 5
Epoch 1/3
1141/1141 [==============================] - 49s 43ms/step - loss: 1.0297e-04
Epoch 2/3
1141/1141 [==============================] - 51s 45ms/step - loss: 1.0297e-04
Epoch 3/3
1141/1141 [==============================] - 59s 51ms/step - loss: 1.0297e-04
Generating test from test index 52095 with words ['vuelta', 'e', 'ir']:
THE COMPLETE RESULTING SENTENCE IS: vueltaeir elípticas.
```

```
La idea es entrenar y probar en un ciclo. Un c
entrenamiento (con un número de epochs) y lueg
a partir del texto y ver si la palabra que da
coloca después de las palabras del texto.

Esto completa un ciclo.

Estos ciclos son acumulativos, y las oraciones
significativas después de cada ciclo sucesivo.

En los hiperparámetros hemos especificado que
teniendo cada uno 3 epochs por ciclo
```

La idea es entrenar y probar en un ciclo. Un ciclo se compone de un sesión de entrenamiento (con un número de epochs) y luego generamos una oración de prueba a partir del texto y ver si la palabra que da la red tiene sentido cuando se coloca después de las palabras del texto.

Esto completa un ciclo.

Estos ciclos son acumulativos, y las oraciones se volverán cada vez más significativas después de cada ciclo sucesivo.

En los hiperparámetros hemos especificado que entrenaremos durante 5 ciclos, teniendo cada uno 3 epochs por ciclo

✓ 13m 8s completed at 3:39 PM