

Anexo 23

Proyecto 23: Clasificador de Texto con RNN's para NLP

Mg. Luis Felipe Bustamante Narváez

En este proyecto, diseñaremos un clasificador de texto, utilizando redes neuronales recurrentes, recurso que permite un procesamiento más complejo de los ejercicios anteriores, pero a su vez más preciso y con excelente eficiencia, y mínimo coste computacional.

Librerías

```
In [... import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Input, GlobalMaxPooling1D
from tensorflow.keras.layers import LSTM, GRU, SimpleRNN, Embedding
from tensorflow.keras.models import Model
from tensorflow.keras.losses import SparseCategoricalCrossentropy
import itertools
from keras.models import load_model
from keras.optimizers import RMSprop
import pickle
```

Cargamos los datos

```
In [... path = 'data/df_total.csv'
df = pd.read_csv(path)
```

In [... df

	url	news	Type
0	https://www.larepublica.co/redirect/post/3201905	Durante el foro La banca articulador empresari...	Otra
1	https://www.larepublica.co/redirect/post/3210288	El regulador de valores de China dijo el domin...	Regulaciones
2	https://www.larepublica.co/redirect/post/3240676	En una industria históricamente masculina como...	Alianzas
3	https://www.larepublica.co/redirect/post/3342889	Con el dato de marzo el IPC interanual encaden...	Macroeconomía
4	https://www.larepublica.co/redirect/post/3427208	Ayer en Cartagena se dio inicio a la versión n...	Otra
...
1212	https://www.bbva.com/es/como-lograr-que-los-in...	En la vida de toda empresa emergente llega un ...	Innovación
1213	https://www.bbva.com/es/podcast-como-nos-afect...	La espiral alcista de los precios continúa y g...	Macroeconomía
1214	https://www.larepublica.co/redirect/post/3253735	Las grandes derrotas nacionales son experienci...	Alianzas
1215	https://www.bbva.com/es/bbva-y-barcelona-healt...	BBVA ha alcanzado un acuerdo de colaboración c...	Innovación
1216	https://www.larepublica.co/redirect/post/3263980	Casi entrando a la parte final de noviembre la...	Alianzas

1217 rows × 3 columns

Procesamiento de Datos

Creamos las categorías

```
In [... target = df['Type'].astype('category').cat.codes
```

```
In [... target
```

```
Out[... 0      3
      1      4
      2      0
      3      2
      4      3
      ..
     1212     1
     1213     2
     1214     0
     1215     1
     1216     0
     Length: 1217, dtype: int8
```

```
In [... # Adicionamos la columna al df
df['target'] = target
```

```
In [... df
```

```
Out[...      url      news      Type  target
0  https://www.larepublica.co/redirect/post/3201905  Durante el foro La banca articulador empresari...      Otra      3
1  https://www.larepublica.co/redirect/post/3210288  El regulador de valores de China dijo el domin...  Regulaciones      4
2  https://www.larepublica.co/redirect/post/3240676  En una industria históricamente masculina como...  Alianzas      0
3  https://www.larepublica.co/redirect/post/3342889  Con el dato de marzo el IPC interanual encaden...  Macroeconomía      2
4  https://www.larepublica.co/redirect/post/3427208  Ayer en Cartagena se dio inicio a la versión n...      Otra      3
...      ...      ...      ...      ...
1212  https://www.bbva.com/es/como-lograr-que-los-in...  En la vida de toda empresa emergente llega un ...  Innovación      1
1213  https://www.bbva.com/es/podcast-como-nos-afect...  La espiral alcista de los precios continúa y g...  Macroeconomía      2
1214  https://www.larepublica.co/redirect/post/3253735  Las grandes derrotas nacionales son experienci...  Alianzas      0
1215  https://www.bbva.com/es/bbva-y-barcelona-healt...  BBVA ha alcanzado un acuerdo de colaboración c...  Innovación      1
1216  https://www.larepublica.co/redirect/post/3263980  Casi entrando a la parte final de noviembre la...  Alianzas      0
```

1217 rows x 4 columns

Separamos los conjuntos de Datos

```
In [... df_train, df_test = train_test_split(df, test_size=0.3)
```

Obtenemos el número de clases

```
In [... K = df['target'].max() + 1
K
```

```
Out[... 7
```

Creamos los conjuntos de salida

```
In [... Y_train = df_train['target']
Y_test = df_test['target']
```

Tokenización

Tokenizamos oraciones en secuencias

```
In [... #Vocabulario máximo
max_vocab_size = 30000
#Iniciamos el tokenizador
tokenizer = Tokenizer(num_words=max_vocab_size)
#Tokenizamos
tokenizer.fit_on_texts(df_train['news'])
#Creamos las secuencias
sequences_train = tokenizer.texts_to_sequences(df_train['news'])
sequences_test = tokenizer.texts_to_sequences(df_test['news'])
```

Diccionario de palabras tokenizadas

```
In [... # Creamos el diccionario
word2index = tokenizer.word_index
# Calculamos el tamaño del tokenizado
V = len(word2index)
# mostramos
print(f'Se encontraron {V} tokens.')
```

Se encontraron 26213 tokens.

```
In [... diez = dict(itertools.islice(word2index.items(), 10))
print(f'Estas son las 10 primeras palabras que más se repiten son:\n{diez}')
```

Estas son las 10 primeras palabras que más se repiten son:

```
{'de': 1, 'la': 2, 'en': 3, 'el': 4, 'que': 5, 'y': 6, 'a': 7, 'los': 8, 'las': 9, 'del': 10}
```

Rellenamos las Secuencias (padding)

```
In [... # Rellenar la secuencia de entrenamiento
data_train = pad_sequences(sequences_train)
print(f'Dimensiones del tensor de entrenamiento: {data_train.shape}')
# Longitud de la secuencia de entrenamiento
T = data_train.shape[1]
print(f'Longitud de la secuencia de entrenamiento: {T}')
```

Dimensiones del tensor de entrenamiento: (851, 3015)

Longitud de la secuencia de entrenamiento: 3015

```
In [... # Rellenar la secuencia de prueba
data_test = pad_sequences(sequences_test, maxlen=T)
print(f'Dimensiones del tensor de prueba: {data_test.shape}')
# Longitud de la secuencia de prueba
print(f'Longitud de la secuencia de prueba: {data_test.shape[1]}')
```

Dimensiones del tensor de prueba: (366, 3015)

Longitud de la secuencia de prueba: 3015

Embedding y Modelo

Dimensiones del Embedding

```
In [... D = 20
```

Construcción del Modelo

```
In [... # Capa de entrada
i = Input(shape=(T,))
# Capa de embedding
x = Embedding(V + 1, D)(i) ##+1 para el token especial de palabras desconocidas padding
# Capa de convolución
x = LSTM(32, return_sequences=True)(x) # 32 filtros para las secuencias de palabras
# Capa de pooling
x = GlobalMaxPooling1D()(x)
# Capa Densa
x = Dense(K)(x)
# Creación del modelo
modelo = Model(i, x)
```

Resumen del Modelo

```
In [... modelo.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 3015)	0
embedding (Embedding)	(None, 3015, 20)	524,280
lstm (LSTM)	(None, 3015, 32)	6,784
global_max_pooling1d (GlobalMaxPooling1D)	(None, 32)	0
dense (Dense)	(None, 7)	231

Total params: 531,295 (2.03 MB)

Trainable params: 531,295 (2.03 MB)

Non-trainable params: 0 (0.00 B)

Compilamos el Modelo

```
In [... modelo.compile(
    loss= SparseCategoricalCrossentropy(from_logits=True),
    optimizer='adam',
    metrics=['accuracy']
)
```

Entrenamos el Modelo

```
In [... print('Entrenando el modelo...')
r = modelo.fit(
    data_train,
```

```
Y_train,  
epochs=50,  
validation_data=(data_test, Y_test)  
)
```

Gráfico de la pérdida por iteración

```
In [ ... # Gráfico de la función de pérdida (loss)  
plt.plot(r.history['loss'], label='Pérdida del Entrenamiento', color='blue')  
plt.plot(r.history['val_loss'], label='Pérdida del set de prueba', color='orange')  
plt.xlabel('Épocas')  
plt.ylabel('Pérdida')  
plt.title('Evolución del entrenamiento')  
plt.legend()  
plt.grid(True)  
plt.show()
```

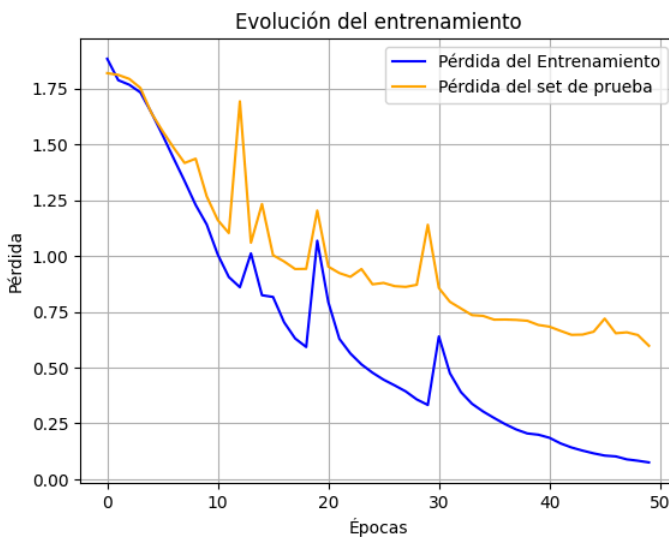
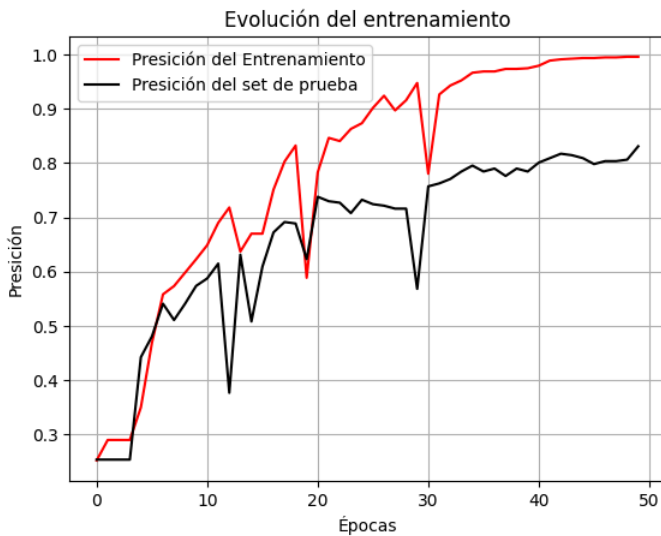


Gráfico de la precisión por iteración

```
In [ ... # Gráfico de la métrica de precisión (accuracy)  
plt.plot(r.history['accuracy'], label='Precisión del Entrenamiento', color='red')  
plt.plot(r.history['val_accuracy'], label='Precisión del set de prueba', color='black')  
plt.xlabel('Épocas')  
plt.ylabel('Precisión')  
plt.title('Evolución del entrenamiento')  
plt.legend()  
plt.grid(True)  
plt.show()
```



Guardar Modelo y otros archivos necesarios

```
In [... # Archivo con extensión HDF5 (deprecado) o keras (actual)
modelo.save('modelo23.keras')
print('Modelo guardado con éxito.')
```

Modelo guardado con éxito.

```
In [... # Guardamos los pesos
modelo.save_weights('modelo23_pesos.weights.h5')
print('Pesos del modelo guardados con éxito.')
```

Pesos del modelo guardados con éxito.

```
In [... # Guardamos el historial del modelo
with open('historial_entrenamiento_23.pkl', 'wb') as f:
    pickle.dump(r.history, f)
```

```
In [... # Guardamos el tokenizador
with open('tokenizer_23.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)
```

Cargar el Modelo para futuras pruebas

```
In [... model_load = load_model('modelo23.keras', compile=False)

model_load.compile(
    optimizer = RMSprop(),
    loss = SparseCategoricalCrossentropy(from_logits=True),
    metrics = ['accuracy']
)

print(f'El Modelo {model_load} se ha cargado, y recompilado correctamente.')
```

El Modelo <Functional name=functional, built=True> se ha cargado, y recompilado correctamente.

Probamos el Modelo con Datos Nuevos

Función para predecir texto

```
In [... def predecir_texto(texto, modelo, tokenizer, T, idx2label=None):
    # Asegurarse que el texto está en una lista
    if isinstance(texto, str):
        texto = [texto]

    # Tokenizar y hacer padding
    secuencia = tokenizer.texts_to_sequences(texto)
    secuencia_padded = pad_sequences(secuencia, maxlen=T)

    # Predicción
    pred = modelo.predict(secuencia_padded)
    clase_predicha = np.argmax(pred, axis=1)[0]

    # Mostrar resultado
    if idx2label:
        print(f'Clase predicha: {clase_predicha} ({idx2label[clase_predicha]})')
        return idx2label[clase_predicha]
    else:
        print(f'Clase predicha: {clase_predicha}')
        return clase_predicha
```

Llamamos la función

```
In [... texto_de_prueba = "Este es un ejemplo de noticia económica internacional"
# Crear mapeo inverso de índices a nombres
idx2label = dict(enumerate(df['Type'].astype('category').cat.categories))
predecir_texto(texto_de_prueba, model_load, tokenizer, T)
idx2label
```

1/1 ————— 1s 833ms/step

Clase predicha: 4

```
Out[... {0: 'Alianzas',
1: 'Innovacion',
2: 'Macroeconomia',
3: 'Otra',
4: 'Regulaciones',
5: 'Reputacion',
6: 'Sostenibilidad'}
```

Conclusiones

En este modelo, se logró crear una clasificación a través de keras, capaz de identificar, a partir del contenido de una noticia, cuál es su categoría. Con un entrenamiento a través de embeddings y redes neuronales recurrentes, hemos generado un clasificador de mayor precisión capaz de clasificar cualquier texto informativo. Aunque el tiempo de entrenamiento es mucho mayor, la precisión del modelo es bastante mejor que los anteriores.