Análisis de Datos y Aprendizaje Máquina con Tensorflow 2.0: Redes neuronales recurrentes

2019/09/30

1 RNN - Clasificación de Texto

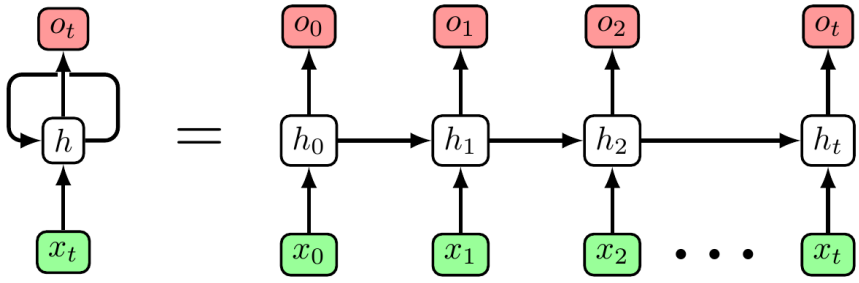
- Objetivo: Implementar RNN y LSTM para la clasificación de secuencias (texto). Aplicar inicialización de pesos y dropout en modelos recurrentes. Se comparará una RNN profunda con una LSTM de una capa.
- Conocer efectos de inicialización y dropout en RNN y comparar con MLP
- Conocer diferentes implementaciones de RNN

1.1 Redes Neuronales Recurrentes

- Cada capa en cada iteración comparte parámetros
- 'sparse_categorical_crossentropy' se utiliza para varias clases

1.2 Leer Dataset

```
In [1]: from tensorflow.keras.models import Model
        from tensorflow.keras.layers import LSTM, Embedding, Dense, SimpleRNN
        from tensorflow.keras.datasets import reuters
        from tensorflow.keras.models import Sequential
        from tensorflow import keras
```



RNN

```
import numpy as np
from tensorflow.keras.preprocessing.sequence import pad_sequences
import matplotlib.pyplot as plt
```

1.3 11,228 noticias de Reuters, con más de 46 temas.

In [2]:
```
# parámetros
emb_dim = 256
num_words = 10000
max_words = 100
```

In [3]:
```
(x_train, y_train), (x_test, y_test) = reuters.load_data(num_words = num_words, maxlen=ma
```

1.4 Temas

In [4]:
```
y_test[:1000]
```

Out[4]:
```
array([23,  3, 19,  3,  3,  3,  9,  3,  3,  3,  1, 19,  4, 40,  1,  4,  3,
        4,  3, 20,  3,  4, 20,  4,  3,  4,  4,  4,  3,  3,  3, 21, 16,  3,
        2,  3, 39,  4,  3,  3,  3,  1,  3,  3,  4,  3,  3,  3,  1, 20,  3,
        3,  4,  3,  4,  4,  3,  4,  3,  4, 19,  4, 18,  3, 19,  3,  3, 19,
        3,  3, 23,  4,  3,  3,  3,  3,  4,  4,  3, 11,  3, 41,  3,  3,  2,
        4,  3, 10,  3,  3,  3, 18,  3,  3, 19,  3,  3,  3,  3,  3, 36,  8,
        3,  4,  4, 19,  3,  4,  4, 19,  2,  3,  4,  3,  3,  3,  4,  3,  3,
        3, 13,  3,  3,  4, 19,  1,  3,  3, 13,  4,  4,  3,  3, 19,  3,  4,
        3,  3,  4, 26,  4,  4,  3,  4,  4,  3,  3, 20,  3, 19,  4,  3,  3,
        3, 10,  3, 25,  1,  4,  4,  4,  3,  6,  3,  3,  3,  4, 35,  3,  3,
        3,  3,  4,  4,  3,  3, 26, 24,  3,  4,  3,  4,  3, 17,  4,  3,  3,
       43,  3,  4, 10, 21,  3,  3,  3, 38,  3,  3,  3, 16,  3,  4,  3, 44,
        3,  3, 16,  4,  3,  3,  3,  4,  3,  3,  4,  3,  1, 24,  3, 41,  3,
        3,  3,  3, 19,  3,  4,  3,  8,  3, 12, 19, 19, 34,  3,  4,  3,  3,
        4,  4,  4,  4,  3,  4,  4,  3,  4,  3,  3,  3,  3,  3, 40,  4,  3,
        3,  3,  3,  4,  3,  3,  3,  4,  1,  4,  4,  3,  4,  4,  4,  1, 41,
        3,  3,  4,  4, 19, 13, 44,  4, 13,  4, 20,  3,  2,  3,  4,  4,  3,
       31,  3,  3, 19,  3,  3,  3,  3,  4,  3,  3,  3,  4,  3,  3,  3, 28,
       19,  3,  3,  3,  3, 18,  3,  4,  3,  4, 19, 20,  3,  3,  0,  4,  3,
        3, 39,  3, 19,  3,  4,  3,  3,  3, 35,  3,  3,  3,  3,  3,  3,  3,
        4,  3,  3,  1,  1,  4, 22,  4,  3, 12,  3,  3, 23,  3,  4,  4,  3,
        4,  3,  3,  1, 12,  3,  3,  3,  3,  3,  3,  3, 20,  1, 28,  4,  3,
        3,  3,  3,  8,  3,  3,  3, 40,  4,  4,  4,  3,  3, 19,  3, 30,  3,
       16,  3,  3,  3,  4,  3,  3,  3, 16,  3,  3,  4,  4,  5,  3,  3,  3,
        3,  3,  4,  3, 10,  3,  3,  3,  3,  3,  1, 13,  3,  4,  3,  4,  4,
        3,  3,  3, 20, 23, 32,  3,  3,  3, 12,  3,  3,  3,  4,  3, 31,  3,
        3,  3, 41,  9,  3,  8,  4,  3,  3, 24, 10, 16, 19,  3,  3,  3,  3,
        4,  4, 23,  3,  3,  3,  7,  3,  8,  8, 19, 20,  3,  4,  3,  4, 33,
        3,  3,  3,  4,  3,  3,  3,  3,  3,  3, 25,  5,  4, 24,  4,  3,  4,
        3, 25,  3,  3,  3,  3,  4,  3,  3, 18,  3, 19,  3,  4,  3,  3,  3,
        4,  6, 12, 11,  3,  3,  4,  0,  3, 15, 19,  3,  4,  3,  4,  3, 20,
        3,  3,  4, 31,  3,  4,  3,  9,  3,  3,  4,  3,  3,  3,  4,  4,  3,
        3, 19,  3, 16, 31,  3,  3,  3,  3,  4,  3, 25,  3,  3,  3,  4, 24,
```

```
 4, 40,  3,  4,  3,  3,  2,  3,  3,  3,  4,  3,  4,  8,  3,  3,  3,
 4,  3,  3,  3,  3,  4,  4,  3, 17,  4,  3,  4, 19,  3,  3,  3,  3,
 4,  5,  3,  3,  3,  8,  3,  4,  3, 13,  3, 20, 33,  3,  3,  3,  6,
20,  8,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  4,  1,  4,  3, 20,
18,  3,  3,  3,  4, 12, 20,  3, 19, 27,  3,  3,  4,  4,  3,  4,  8,
 3,  3,  3,  3,  3,  3,  4,  3,  3,  4,  4,  3,  3,  3,  3,  4, 16,
 8,  3,  4,  4, 40,  4,  3,  4,  4, 16,  3,  1,  4, 19,  3,  3,  3,
 3,  3,  3,  4,  3,  4,  3, 18, 20,  3,  4,  4,  4,  4,  4,  3, 19,
 4, 13,  3,  3,  4,  3,  3, 25, 24,  4,  3,  4,  3,  3,  4,  3, 19,
 3,  4, 19, 16, 32,  3,  3,  4, 30,  3, 13, 19, 11,  1,  3,  3, 11,
19,  3,  4,  3,  4, 24,  3,  4,  3,  4, 19,  4,  4,  1,  3,  3,  4,
21,  3,  4,  3,  4, 19,  4, 17,  3,  1,  3,  3,  3,  3,  3,  4,  3,
 3,  3,  8,  3,  3,  4,  3,  3,  3,  4,  3,  3,  3,  3,  3,  4,  9,
 3,  4,  4, 44,  3,  3,  3, 38,  4, 19, 32, 19,  4, 10,  4, 20,  3,
33,  3, 30, 19,  3,  3,  3,  3,  3, 10,  4,  1,  3,  4,  1,  3,  4,
20,  3,  3,  4, 25,  3,  3,  1,  3,  3,  3, 22, 19,  3,  3,  3,  3,
11,  3,  3,  3,  4,  3, 19,  4,  4,  3,  4,  3,  3,  3,  3,  3,  3,
 1,  3,  3,  1,  4,  3,  4, 33,  4,  4,  3,  3,  4,  3,  3,  3,  3,
 3,  3,  4,  4,  3,  1,  3, 25,  3, 12,  3,  3,  8,  1, 31,  3,  4,
 3,  3,  3,  3,  4,  3,  1,  4,  8,  3,  3, 13,  3,  4,  4,  3,  3,
43,  4, 36, 10, 43,  3, 12,  2,  3,  4,  4,  8,  3,  3,  4,  4,  3,
 4, 19, 16, 16,  1,  4,  4,  3,  4, 16,  3,  4,  4,  3,  3,  3,  3,
19,  2, 18, 20, 13,  3,  3,  3,  3,  3,  3, 16,  4,  4,  3,  4,  4,
16, 11,  3,  4,  4, 11, 24,  3,  3,  3, 29, 16,  1,  3, 20,  4,  3,
13,  3,  3, 24, 32,  4,  4,  3,  3,  3,  4,  3, 44,  4,  3, 20,  3,
 4, 19,  3, 19,  9,  3,  3,  3,  3,  3,  3,  3,  3,  3])
```

In [5]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4777,)
(1195,)
(4777,)
(1195,)
```

In [6]:
```
print('Noticia')
print(x_train[0])
print('Etiqueta')
print(y_train[0])
```

```
Noticia
[1, 2, 2, 8, 43, 10, 447, 5, 25, 207, 270, 5, 3095, 111, 16, 369, 186, 90, 67, 7, 89, 5, 19, 102,
Etiqueta
3
```

1.5 Palabras de noticia

```
In [7]: wordDict = {y:x for x,y in reuters.get_word_index().items()}
        res = []
        for index in x_train[0]:
            res.append(wordDict.get(index - 3))
        print('Noticia: ',res,'Longitud noticia: ', len(res))
```

```
Noticia:  [None, None, None, 'said', 'as', 'a', 'result', 'of', 'its', 'december', 'acquisition',
```

```
In [8]: x_train = pad_sequences(x_train, maxlen=max_words, padding = 'post')
        x_test = pad_sequences(x_test, maxlen=max_words, padding = 'post')
```

```
In [9]: epoch = 40
        verbose = 1
        batch = 50
```

```
In [10]: print(x_train.shape)
         print(x_test.shape)
         print(y_train.shape)
         print(y_test.shape)
```

```
(4777, 100)
(1195, 100)
(4777,)
(1195,)
```

1.6 Deep RNN

- Cuando se conectan varias capas de RNNs se modifica el parámetro 'return_sequences'
- Se inicializan los pesos con 'glorot_uniform'

```
In [11]: def deep_rnn():
             model = Sequential()
             model.add(Embedding(num_words, emb_dim))
             model.add(SimpleRNN(32, return_sequences = True, recurrent_initializer='glorot_unifo
             model.add(SimpleRNN(32, return_sequences = False, recurrent_initializer='glorot_unif
             model.add(Dense(46, activation = 'softmax'))

             model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['ac

             return model
```

```
In [12]: model = deep_rnn()
         model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
embedding (Embedding)        (None, None, 256)         2560000
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, None, 32)          9248
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 32)                2080
_________________________________________________________________
dense (Dense)                (None, 46)                1518
=================================================================
Total params: 2,572,846
Trainable params: 2,572,846
Non-trainable params: 0
_________________________________________________________________
```

```
In [13]: history = model.fit(x_train, y_train, batch_size = batch, validation_split = 0.3,
                             epochs = epoch, verbose = verbose)
```

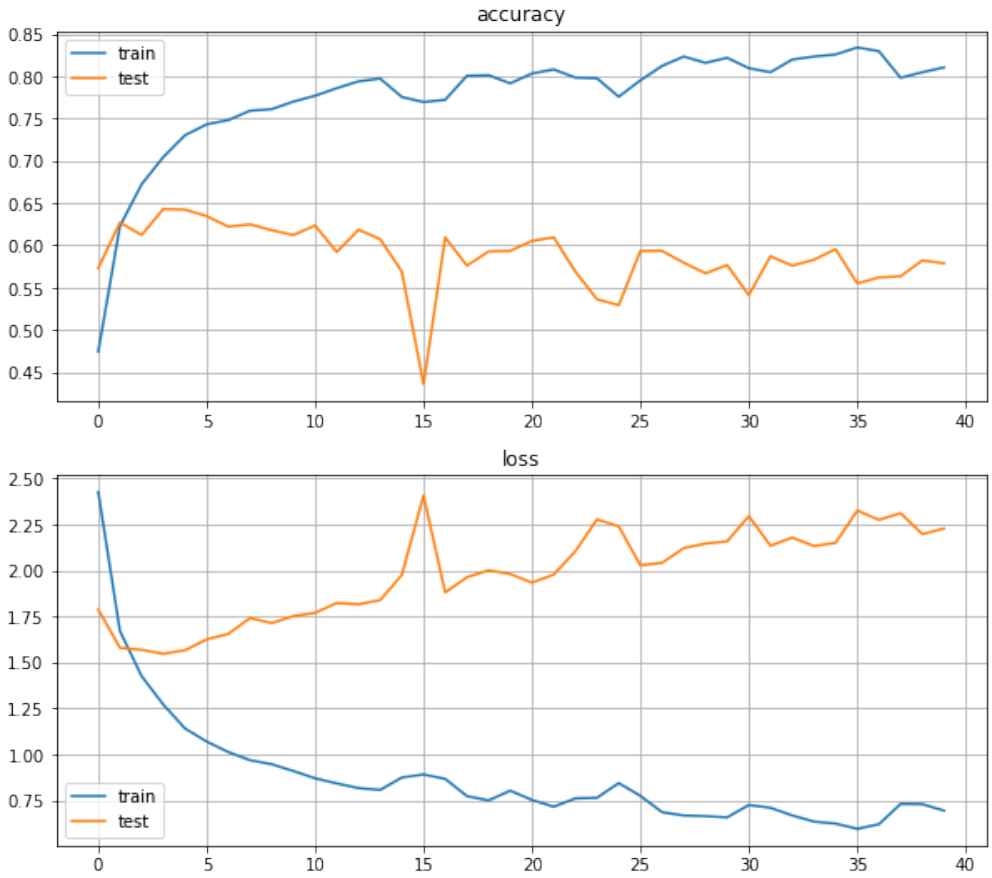
```
Train on 3343 samples, validate on 1434 samples
Epoch 1/40
3343/3343 [==============================] - 9s 3ms/sample - loss: 2.4251 - accuracy: 0.4744 - va
Epoch 2/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 1.6702 - accuracy: 0.6228 - va
Epoch 3/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 1.4253 - accuracy: 0.6722 - va
Epoch 4/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 1.2708 - accuracy: 0.7045 - va
Epoch 5/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 1.1412 - accuracy: 0.7302 - va
Epoch 6/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 1.0698 - accuracy: 0.7430 - va
Epoch 7/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 1.0126 - accuracy: 0.7481 - va
Epoch 8/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.9689 - accuracy: 0.7592 - va
Epoch 9/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.9469 - accuracy: 0.7610 - va
Epoch 10/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.9099 - accuracy: 0.7700 - va
Epoch 11/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8705 - accuracy: 0.7768 - va
Epoch 12/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8427 - accuracy: 0.7858 - va
Epoch 13/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8171 - accuracy: 0.7939 - va
Epoch 14/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8073 - accuracy: 0.7972 - va
Epoch 15/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8749 - accuracy: 0.7754 - va
Epoch 16/40
```

```
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8914 - accuracy: 0.7694 - va
Epoch 17/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8673 - accuracy: 0.7721 - va
Epoch 18/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7738 - accuracy: 0.8005 - va
Epoch 19/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7507 - accuracy: 0.8011 - va
Epoch 20/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8034 - accuracy: 0.7915 - va
Epoch 21/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7532 - accuracy: 0.8032 - va
Epoch 22/40
3343/3343 [==============================] - 7s 2ms/sample - loss: 0.7162 - accuracy: 0.8080 - va
Epoch 23/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7614 - accuracy: 0.7984 - va
Epoch 24/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7648 - accuracy: 0.7975 - va
Epoch 25/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.8446 - accuracy: 0.7757 - va
Epoch 26/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7751 - accuracy: 0.7951 - va
Epoch 27/40
3343/3343 [==============================] - 7s 2ms/sample - loss: 0.6866 - accuracy: 0.8121 - va
Epoch 28/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6687 - accuracy: 0.8232 - va
Epoch 29/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6654 - accuracy: 0.8157 - va
Epoch 30/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6583 - accuracy: 0.8217 - va
Epoch 31/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7258 - accuracy: 0.8095 - va
Epoch 32/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7104 - accuracy: 0.8047 - va
Epoch 33/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6690 - accuracy: 0.8196 - va
Epoch 34/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6354 - accuracy: 0.8232 - va
Epoch 35/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6245 - accuracy: 0.8256 - va
Epoch 36/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.5961 - accuracy: 0.8340 - va
Epoch 37/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6211 - accuracy: 0.8295 - va
Epoch 38/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7317 - accuracy: 0.7981 - va
Epoch 39/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.7301 - accuracy: 0.8047 - va
Epoch 40/40
3343/3343 [==============================] - 8s 2ms/sample - loss: 0.6952 - accuracy: 0.8104 - va
```

```
In [14]: test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

         print('\nTest acccuracy:', test_acc)
```

```
1195/1 - 1s - loss: 1.9598 - accuracy: 0.5799

Test acccuracy: 0.5799163
```

```
In [15]: #plot
         plt.figure(figsize=(10,9))

         plt.subplot(211)
         plt.plot(history.history['accuracy'])
         plt.plot(history.history['val_accuracy'])
         plt.title('accuracy')
         plt.legend(['train', 'test'])
         plt.grid()
         plt.subplot(212)
         plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('loss')
         plt.legend(['train', 'test'])
         plt.grid()



         plt.show()
```



1.7 LSTM

- Desempeño de LSTM con una capa vs. deep RNN

```
In [16]: def lstm():
             model = Sequential()
             model.add(Embedding(num_words, emb_dim))
             model.add(LSTM(32, return_sequences = False))
             model.add(Dense(46, activation = 'softmax'))

             model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['ac

             return model
```

```
In [17]: model = lstm()
         model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 256)         2560000
_________________________________________________________________
lstm (LSTM)                  (None, 32)                36992
_________________________________________________________________
dense_1 (Dense)              (None, 46)                1518
=================================================================
Total params: 2,598,510
Trainable params: 2,598,510
Non-trainable params: 0
_________________________________________________________________
```

```
In [18]: history = model.fit(x_train, y_train, batch_size = batch, validation_split = 0.3,
                             epochs = epoch, verbose = verbose)
```

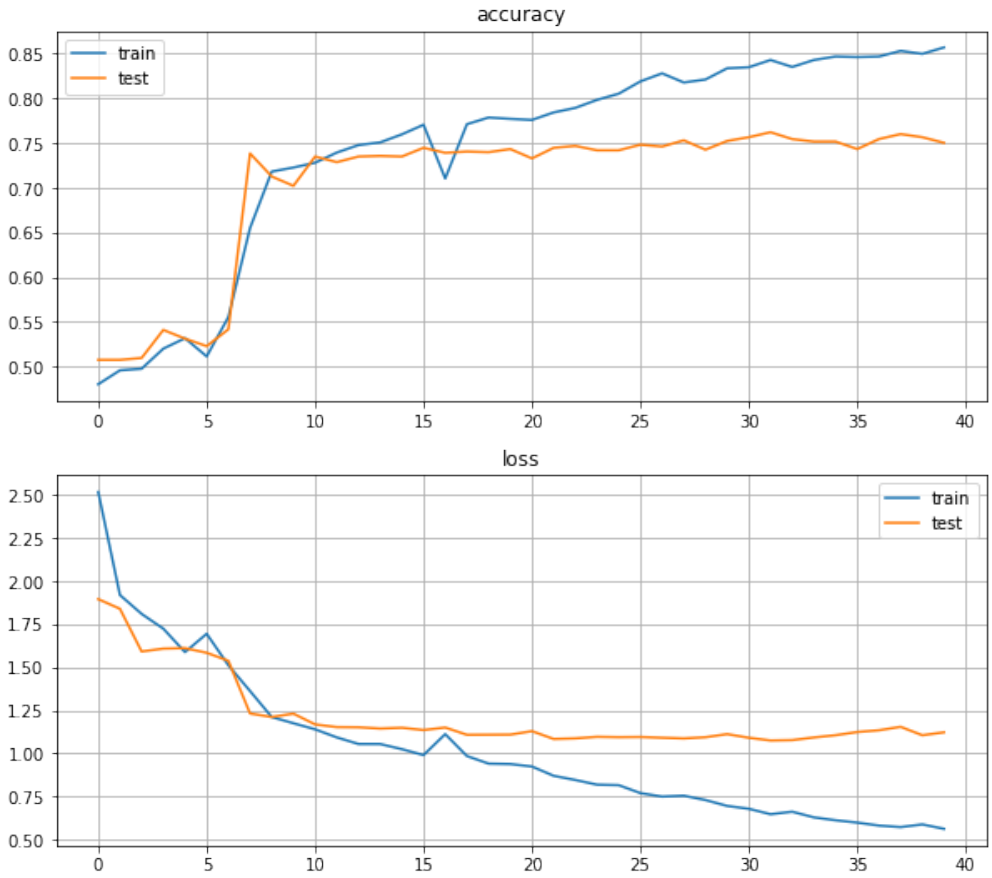
```
Train on 3343 samples, validate on 1434 samples
Epoch 1/40
3343/3343 [==============================] - 5s 1ms/sample - loss: 2.5167 - accuracy: 0.4804 - va
Epoch 2/40
3343/3343 [==============================] - 3s 758us/sample - loss: 1.9191 - accuracy: 0.4960 -
Epoch 3/40
3343/3343 [==============================] - 3s 762us/sample - loss: 1.8095 - accuracy: 0.4978 -
Epoch 4/40
3343/3343 [==============================] - 3s 761us/sample - loss: 1.7238 - accuracy: 0.5202 -
Epoch 5/40
3343/3343 [==============================] - 3s 754us/sample - loss: 1.5878 - accuracy: 0.5319 -
Epoch 6/40
3343/3343 [==============================] - 3s 765us/sample - loss: 1.6942 - accuracy: 0.5115 -
Epoch 7/40
3343/3343 [==============================] - 3s 763us/sample - loss: 1.5115 - accuracy: 0.5555 -
Epoch 8/40
3343/3343 [==============================] - 3s 761us/sample - loss: 1.3620 - accuracy: 0.6551 -
Epoch 9/40
3343/3343 [==============================] - 3s 758us/sample - loss: 1.2112 - accuracy: 0.7182 -
Epoch 10/40
3343/3343 [==============================] - 3s 761us/sample - loss: 1.1750 - accuracy: 0.7227 -
Epoch 11/40
3343/3343 [==============================] - 3s 766us/sample - loss: 1.1398 - accuracy: 0.7281 -
Epoch 12/40
3343/3343 [==============================] - 3s 757us/sample - loss: 1.0929 - accuracy: 0.7395 -
Epoch 13/40
3343/3343 [==============================] - 3s 759us/sample - loss: 1.0547 - accuracy: 0.7478 -
Epoch 14/40
3343/3343 [==============================] - 3s 766us/sample - loss: 1.0545 - accuracy: 0.7508 -
Epoch 15/40
3343/3343 [==============================] - 3s 765us/sample - loss: 1.0248 - accuracy: 0.7598 -
Epoch 16/40
3343/3343 [==============================] - 3s 751us/sample - loss: 0.9903 - accuracy: 0.7706 -
```

```
Epoch 17/40
3343/3343 [==============================] - 3s 763us/sample - loss: 1.1119 - accuracy: 0.7104 -
Epoch 18/40
3343/3343 [==============================] - 3s 770us/sample - loss: 0.9848 - accuracy: 0.7712 -
Epoch 19/40
3343/3343 [==============================] - 3s 757us/sample - loss: 0.9408 - accuracy: 0.7786 -
Epoch 20/40
3343/3343 [==============================] - 3s 758us/sample - loss: 0.9384 - accuracy: 0.7771 -
Epoch 21/40
3343/3343 [==============================] - 3s 761us/sample - loss: 0.9241 - accuracy: 0.7759 -
Epoch 22/40
3343/3343 [==============================] - 3s 756us/sample - loss: 0.8701 - accuracy: 0.7843 -
Epoch 23/40
3343/3343 [==============================] - 3s 759us/sample - loss: 0.8459 - accuracy: 0.7894 -
Epoch 24/40
3343/3343 [==============================] - 3s 759us/sample - loss: 0.8188 - accuracy: 0.7984 -
Epoch 25/40
3343/3343 [==============================] - 3s 764us/sample - loss: 0.8159 - accuracy: 0.8053 -
Epoch 26/40
3343/3343 [==============================] - 3s 764us/sample - loss: 0.7698 - accuracy: 0.8190 -
Epoch 27/40
3343/3343 [==============================] - 3s 769us/sample - loss: 0.7496 - accuracy: 0.8280 -
Epoch 28/40
3343/3343 [==============================] - 3s 762us/sample - loss: 0.7547 - accuracy: 0.8178 -
Epoch 29/40
3343/3343 [==============================] - 3s 760us/sample - loss: 0.7296 - accuracy: 0.8211 -
Epoch 30/40
3343/3343 [==============================] - 3s 763us/sample - loss: 0.6954 - accuracy: 0.8337 -
Epoch 31/40
3343/3343 [==============================] - 3s 760us/sample - loss: 0.6788 - accuracy: 0.8349 -
Epoch 32/40
3343/3343 [==============================] - 3s 757us/sample - loss: 0.6473 - accuracy: 0.8430 -
Epoch 33/40
3343/3343 [==============================] - 3s 758us/sample - loss: 0.6618 - accuracy: 0.8352 -
Epoch 34/40
3343/3343 [==============================] - 3s 763us/sample - loss: 0.6290 - accuracy: 0.8430 -
Epoch 35/40
3343/3343 [==============================] - 3s 762us/sample - loss: 0.6122 - accuracy: 0.8468 -
Epoch 36/40
3343/3343 [==============================] - 3s 758us/sample - loss: 0.5986 - accuracy: 0.8462 -
Epoch 37/40
3343/3343 [==============================] - 3s 756us/sample - loss: 0.5814 - accuracy: 0.8468 -
Epoch 38/40
3343/3343 [==============================] - 3s 762us/sample - loss: 0.5726 - accuracy: 0.8531 -
Epoch 39/40
3343/3343 [==============================] - 3s 762us/sample - loss: 0.5881 - accuracy: 0.8498 -
Epoch 40/40
3343/3343 [==============================] - 3s 765us/sample - loss: 0.5625 - accuracy: 0.8570 -
```

```
In [19]: test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

         print('\nTest acccuracy:', test_acc)
```

```
1195/1 - 0s - loss: 0.9080 - accuracy: 0.7255

Test acccuracy: 0.725523
```

```
In [20]: #plot
         plt.figure(figsize=(10,9))

         plt.subplot(211)
         plt.plot(history.history['accuracy'])
         plt.plot(history.history['val_accuracy'])
         plt.title('accuracy')
         plt.legend(['train', 'test'])
         plt.grid()
         plt.subplot(212)
         plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('loss')
         plt.legend(['train', 'test'])
         plt.grid()



         plt.show()
```



1.8 Deep LSTM

- LSTM cuentan inicializador 'orthogonal' por defecto

```
In [21]: def deep_lstm():
             model = Sequential()
             model.add(Embedding(num_words, emb_dim))
             model.add(LSTM(32, return_sequences = True, recurrent_initializer='orthogonal'))
             model.add(LSTM(32, return_sequences = False, recurrent_initializer='orthogonal'))
             model.add(Dense(46, activation = 'softmax'))
             model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['ac

             return model
```

```
In [22]: model = deep_lstm()
         model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 256)         2560000
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 32)          36992
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_2 (Dense)              (None, 46)                1518
=================================================================
Total params: 2,606,830
Trainable params: 2,606,830
Non-trainable params: 0
_________________________________________________________________
```

```
In [23]: history1 = model.fit(x_train, y_train, batch_size = batch, validation_split = 0.3,
                              epochs = epoch, verbose = verbose)
```

```
Train on 3343 samples, validate on 1434 samples
Epoch 1/40
3343/3343 [==============================] - 6s 2ms/sample - loss: 2.4989 - accuracy: 0.4855 - va
Epoch 2/40
3343/3343 [==============================] - 3s 831us/sample - loss: 1.9301 - accuracy: 0.4957 -
Epoch 3/40
3343/3343 [==============================] - 3s 833us/sample - loss: 1.9158 - accuracy: 0.4957 -
Epoch 4/40
3343/3343 [==============================] - 3s 827us/sample - loss: 1.6895 - accuracy: 0.5878 -
Epoch 5/40
3343/3343 [==============================] - 3s 819us/sample - loss: 1.4778 - accuracy: 0.6476 -
Epoch 6/40
3343/3343 [==============================] - 3s 817us/sample - loss: 1.3521 - accuracy: 0.6683 -
Epoch 7/40
3343/3343 [==============================] - 3s 823us/sample - loss: 1.2462 - accuracy: 0.6850 -
Epoch 8/40
3343/3343 [==============================] - 3s 821us/sample - loss: 1.4841 - accuracy: 0.6096 -
Epoch 9/40
3343/3343 [==============================] - 3s 819us/sample - loss: 1.2544 - accuracy: 0.6710 -
Epoch 10/40
3343/3343 [==============================] - 3s 825us/sample - loss: 1.2162 - accuracy: 0.6778 -
Epoch 11/40
3343/3343 [==============================] - 3s 829us/sample - loss: 1.1861 - accuracy: 0.6844 -
Epoch 12/40
3343/3343 [==============================] - 3s 818us/sample - loss: 1.1697 - accuracy: 0.6907 -
Epoch 13/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.2366 - accuracy: 0.6826 -
Epoch 14/40
3343/3343 [==============================] - 3s 824us/sample - loss: 1.1662 - accuracy: 0.6883 -
Epoch 15/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.1370 - accuracy: 0.6955 -
```

```
Epoch 16/40
3343/3343 [==============================] - 3s 822us/sample - loss: 1.1259 - accuracy: 0.6976 -
Epoch 17/40
3343/3343 [==============================] - 3s 821us/sample - loss: 1.1091 - accuracy: 0.6991 -
Epoch 18/40
3343/3343 [==============================] - 3s 822us/sample - loss: 1.1014 - accuracy: 0.7045 -
Epoch 19/40
3343/3343 [==============================] - 3s 821us/sample - loss: 1.0894 - accuracy: 0.7089 -
Epoch 20/40
3343/3343 [==============================] - 3s 831us/sample - loss: 1.0735 - accuracy: 0.7122 -
Epoch 21/40
3343/3343 [==============================] - 3s 828us/sample - loss: 1.0675 - accuracy: 0.7284 -
Epoch 22/40
3343/3343 [==============================] - 3s 831us/sample - loss: 1.0587 - accuracy: 0.7257 -
Epoch 23/40
3343/3343 [==============================] - 3s 827us/sample - loss: 1.0160 - accuracy: 0.7439 -
Epoch 24/40
3343/3343 [==============================] - 3s 823us/sample - loss: 0.9511 - accuracy: 0.7631 -
Epoch 25/40
3343/3343 [==============================] - 3s 827us/sample - loss: 0.9274 - accuracy: 0.7709 -
Epoch 26/40
3343/3343 [==============================] - 3s 824us/sample - loss: 0.9101 - accuracy: 0.7846 -
Epoch 27/40
3343/3343 [==============================] - 3s 821us/sample - loss: 0.8865 - accuracy: 0.7831 -
Epoch 28/40
3343/3343 [==============================] - 3s 826us/sample - loss: 0.8933 - accuracy: 0.7867 -
Epoch 29/40
3343/3343 [==============================] - 3s 830us/sample - loss: 0.8757 - accuracy: 0.7900 -
Epoch 30/40
3343/3343 [==============================] - 3s 824us/sample - loss: 0.8476 - accuracy: 0.7945 -
Epoch 31/40
3343/3343 [==============================] - 3s 823us/sample - loss: 0.8088 - accuracy: 0.8077 -
Epoch 32/40
3343/3343 [==============================] - 3s 827us/sample - loss: 0.8062 - accuracy: 0.8035 -
Epoch 33/40
3343/3343 [==============================] - 3s 829us/sample - loss: 0.9701 - accuracy: 0.7616 -
Epoch 34/40
3343/3343 [==============================] - 3s 823us/sample - loss: 0.8580 - accuracy: 0.7885 -
Epoch 35/40
3343/3343 [==============================] - 3s 824us/sample - loss: 0.8016 - accuracy: 0.8041 -
Epoch 36/40
3343/3343 [==============================] - 3s 822us/sample - loss: 0.7805 - accuracy: 0.8115 -
Epoch 37/40
3343/3343 [==============================] - 3s 832us/sample - loss: 0.7519 - accuracy: 0.8139 -
Epoch 38/40
3343/3343 [==============================] - 3s 824us/sample - loss: 0.7465 - accuracy: 0.8169 -
Epoch 39/40
3343/3343 [==============================] - 3s 827us/sample - loss: 0.7130 - accuracy: 0.8220 -
Epoch 40/40
```

```
3343/3343 [==============================] - 3s 825us/sample - loss: 0.7012 - accuracy: 0.8277 -
```

```
In [24]: test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

         print('\nTest acccuracy:', test_acc)
```

```
1195/1 - 0s - loss: 1.1259 - accuracy: 0.7364

Test acccuracy: 0.7364017
```

- Modificando inicializador

```
In [25]: def deep_lstm():
             model = Sequential()
             model.add(Embedding(num_words, emb_dim))
             model.add(LSTM(32, return_sequences = True, recurrent_initializer='glorot_uniform'))
             model.add(LSTM(32, return_sequences = False, recurrent_initializer='glorot_uniform')
             model.add(Dense(46, activation = 'softmax'))
             model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['ac

             return model
```

```
In [26]: model = deep_lstm()
         model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 256)         2560000
_________________________________________________________________
lstm_3 (LSTM)                (None, None, 32)          36992
_________________________________________________________________
lstm_4 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_3 (Dense)              (None, 46)                1518
=================================================================
Total params: 2,606,830
Trainable params: 2,606,830
Non-trainable params: 0
_________________________________________________________________
```

```
In [27]: history2 = model.fit(x_train, y_train, batch_size = batch, validation_split = 0.3,
                              epochs = 40, verbose = verbose)
```

```
Train on 3343 samples, validate on 1434 samples
Epoch 1/40
3343/3343 [==============================] - 6s 2ms/sample - loss: 2.5700 - accuracy: 0.4762 - va
```

```
Epoch 2/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.9325 - accuracy: 0.4957 -
Epoch 3/40
3343/3343 [==============================] - 3s 831us/sample - loss: 1.9176 - accuracy: 0.4957 -
Epoch 4/40
3343/3343 [==============================] - 3s 818us/sample - loss: 1.9162 - accuracy: 0.4957 -
Epoch 5/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.9131 - accuracy: 0.4957 -
Epoch 6/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.6857 - accuracy: 0.5890 -
Epoch 7/40
3343/3343 [==============================] - 3s 824us/sample - loss: 1.4675 - accuracy: 0.6730 -
Epoch 8/40
3343/3343 [==============================] - 3s 819us/sample - loss: 1.3910 - accuracy: 0.6862 -
Epoch 9/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.3025 - accuracy: 0.7015 -
Epoch 10/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.1942 - accuracy: 0.7203 -
Epoch 11/40
3343/3343 [==============================] - 3s 813us/sample - loss: 1.1409 - accuracy: 0.7329 -
Epoch 12/40
3343/3343 [==============================] - 3s 808us/sample - loss: 1.1163 - accuracy: 0.7451 -
Epoch 13/40
3343/3343 [==============================] - 3s 815us/sample - loss: 1.0596 - accuracy: 0.7526 -
Epoch 14/40
3343/3343 [==============================] - 3s 826us/sample - loss: 1.0246 - accuracy: 0.7586 -
Epoch 15/40
3343/3343 [==============================] - 3s 845us/sample - loss: 1.0034 - accuracy: 0.7667 -
Epoch 16/40
3343/3343 [==============================] - 3s 832us/sample - loss: 0.9865 - accuracy: 0.7727 -
Epoch 17/40
3343/3343 [==============================] - 3s 828us/sample - loss: 0.9736 - accuracy: 0.7718 -
Epoch 18/40
3343/3343 [==============================] - 3s 820us/sample - loss: 1.0098 - accuracy: 0.7562 -
Epoch 19/40
3343/3343 [==============================] - 3s 820us/sample - loss: 0.9692 - accuracy: 0.7625 -
Epoch 20/40
3343/3343 [==============================] - 3s 821us/sample - loss: 0.9559 - accuracy: 0.7616 -
Epoch 21/40
3343/3343 [==============================] - 3s 825us/sample - loss: 0.9282 - accuracy: 0.7658 -
Epoch 22/40
3343/3343 [==============================] - 3s 821us/sample - loss: 0.9061 - accuracy: 0.7751 -
Epoch 23/40
3343/3343 [==============================] - 3s 817us/sample - loss: 0.8745 - accuracy: 0.7846 -
Epoch 24/40
3343/3343 [==============================] - 3s 822us/sample - loss: 0.8889 - accuracy: 0.7828 -
Epoch 25/40
3343/3343 [==============================] - 3s 825us/sample - loss: 0.8368 - accuracy: 0.7966 -
Epoch 26/40
```

```
3343/3343 [==============================] - 3s 824us/sample - loss: 0.8160 - accuracy: 0.7975 -
Epoch 27/40
3343/3343 [==============================] - 3s 815us/sample - loss: 0.7953 - accuracy: 0.8002 -
Epoch 28/40
3343/3343 [==============================] - 3s 821us/sample - loss: 0.7865 - accuracy: 0.7987 -
Epoch 29/40
3343/3343 [==============================] - 3s 816us/sample - loss: 0.8294 - accuracy: 0.7891 -
Epoch 30/40
3343/3343 [==============================] - 3s 819us/sample - loss: 0.7485 - accuracy: 0.8124 -
Epoch 31/40
3343/3343 [==============================] - 3s 833us/sample - loss: 0.7390 - accuracy: 0.8109 -
Epoch 32/40
3343/3343 [==============================] - 3s 832us/sample - loss: 0.7400 - accuracy: 0.8121 -
Epoch 33/40
3343/3343 [==============================] - 3s 823us/sample - loss: 0.7524 - accuracy: 0.8136 -
Epoch 34/40
3343/3343 [==============================] - 3s 819us/sample - loss: 0.7107 - accuracy: 0.8175 -
Epoch 35/40
3343/3343 [==============================] - 3s 825us/sample - loss: 0.6866 - accuracy: 0.8256 -
Epoch 36/40
3343/3343 [==============================] - 3s 818us/sample - loss: 0.6601 - accuracy: 0.8316 -
Epoch 37/40
3343/3343 [==============================] - 3s 821us/sample - loss: 0.6462 - accuracy: 0.8358 -
Epoch 38/40
3343/3343 [==============================] - 3s 826us/sample - loss: 0.6195 - accuracy: 0.8471 -
Epoch 39/40
3343/3343 [==============================] - 3s 819us/sample - loss: 0.6402 - accuracy: 0.8403 -
Epoch 40/40
3343/3343 [==============================] - 3s 813us/sample - loss: 0.7507 - accuracy: 0.8163 -
```

```
In [28]: test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

         print('\nTest acccuracy:', test_acc)

1195/1 - 0s - loss: 0.8521 - accuracy: 0.7506

Test acccuracy: 0.75062764
```

- Recurrent dropout

```
In [39]: def deep_lstm():
             model = Sequential()
             model.add(Embedding(num_words, emb_dim))
             model.add(LSTM(32, return_sequences = True, recurrent_initializer='glorot_uniform',
                            recurrent_dropout=0.05))
             model.add(LSTM(32, return_sequences = False, recurrent_initializer='glorot_uniform',
                            recurrent_dropout=0.05))
             model.add(Dense(46, activation = 'softmax'))
```

```
            model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['ac

            return model
```

```
In [40]: model = deep_lstm()
         model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_6 (Embedding)      (None, None, 256)         2560000
_________________________________________________________________
lstm_9 (LSTM)                (None, None, 32)          36992
_________________________________________________________________
lstm_10 (LSTM)               (None, 32)                8320
_________________________________________________________________
dense_6 (Dense)              (None, 46)                1518
=================================================================
Total params: 2,606,830
Trainable params: 2,606,830
Non-trainable params: 0
_________________________________________________________________
```

```
In [41]: history3 = model.fit(x_train, y_train, batch_size = batch, validation_split = 0.3,
                              epochs = epoch, verbose = verbose)
```

```
Train on 3343 samples, validate on 1434 samples
Epoch 1/40
3343/3343 [==============================] - 15s 5ms/sample - loss: 2.5300 - accuracy: 0.4807 - v
Epoch 2/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.7890 - accuracy: 0.5423 - v
Epoch 3/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.5052 - accuracy: 0.6473 - v
Epoch 4/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.4009 - accuracy: 0.6512 - v
Epoch 5/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.2882 - accuracy: 0.6838 - v
Epoch 6/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.2742 - accuracy: 0.6847 - v
Epoch 7/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.2382 - accuracy: 0.6847 - v
Epoch 8/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.1818 - accuracy: 0.7048 - v
Epoch 9/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.1714 - accuracy: 0.7170 - v
Epoch 10/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.1620 - accuracy: 0.7326 - v
Epoch 11/40
```

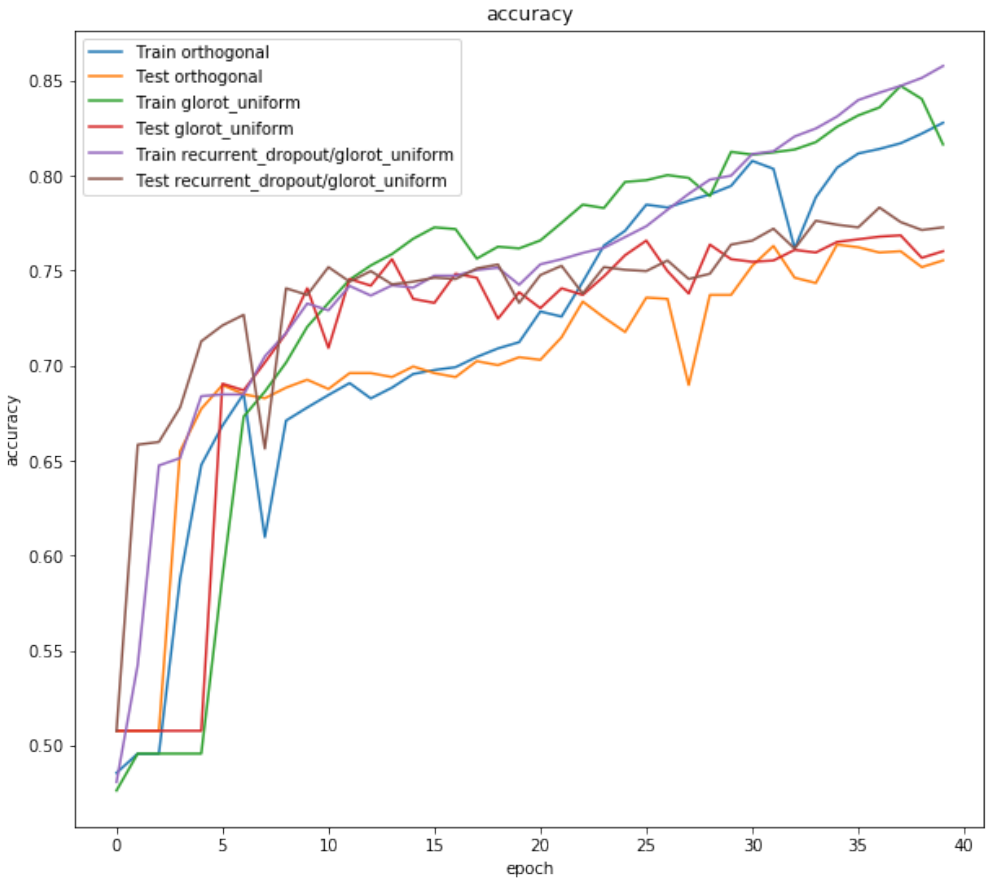
```
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.1490 - accuracy: 0.7290 - v
Epoch 12/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.1083 - accuracy: 0.7418 - v
Epoch 13/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.1040 - accuracy: 0.7368 - v
Epoch 14/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.0769 - accuracy: 0.7421 - v
Epoch 15/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.0798 - accuracy: 0.7410 - v
Epoch 16/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.0545 - accuracy: 0.7472 - v
Epoch 17/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.0386 - accuracy: 0.7472 - v
Epoch 18/40
3343/3343 [==============================] - 14s 4ms/sample - loss: 1.0312 - accuracy: 0.7502 - v
Epoch 19/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.0085 - accuracy: 0.7514 - v
Epoch 20/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 1.0313 - accuracy: 0.7424 - v
Epoch 21/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.9799 - accuracy: 0.7532 - v
Epoch 22/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.9604 - accuracy: 0.7559 - v
Epoch 23/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.9553 - accuracy: 0.7592 - v
Epoch 24/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.9172 - accuracy: 0.7619 - v
Epoch 25/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.8817 - accuracy: 0.7676 - v
Epoch 26/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.8580 - accuracy: 0.7733 - v
Epoch 27/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.8405 - accuracy: 0.7819 - v
Epoch 28/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.8099 - accuracy: 0.7903 - v
Epoch 29/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.7923 - accuracy: 0.7978 - v
Epoch 30/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.7696 - accuracy: 0.7999 - v
Epoch 31/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.7486 - accuracy: 0.8112 - v
Epoch 32/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.7392 - accuracy: 0.8127 - v
Epoch 33/40
3343/3343 [==============================] - 14s 4ms/sample - loss: 0.7216 - accuracy: 0.8205 - v
Epoch 34/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.7020 - accuracy: 0.8247 - v
Epoch 35/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.6766 - accuracy: 0.8310 - v
```

```
Epoch 36/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.6432 - accuracy: 0.8397 - v
Epoch 37/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.6359 - accuracy: 0.8436 - v
Epoch 38/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.6184 - accuracy: 0.8471 - v
Epoch 39/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.6000 - accuracy: 0.8513 - v
Epoch 40/40
3343/3343 [==============================] - 13s 4ms/sample - loss: 0.5776 - accuracy: 0.8576 - v
```

```
In [42]: test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)

         print('\nTest acccuracy:', test_acc)
```

```
1195/1 - 2s - loss: 0.7249 - accuracy: 0.7556

Test acccuracy: 0.75564855
```

```
In [43]: #plot
         plt.figure(figsize=(10,9))
         plt.plot(history1.history['accuracy'])
         plt.plot(history1.history['val_accuracy'])


         plt.plot(history2.history['accuracy'])
         plt.plot(history2.history['val_accuracy'])

         plt.plot(history3.history['accuracy'])
         plt.plot(history3.history['val_accuracy'])


         plt.legend(['Train orthogonal', 'Test orthogonal',
                     'Train glorot_uniform', 'Test glorot_uniform',
                     'Train recurrent_dropout/glorot_uniform', 'Test recurrent_dropout/glorot_uni

         plt.title('accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')

         plt.show()
```



- Mejorar test accuracy
- Investigar que son las GRU e implementarlas
- Experimentar con diferente número de capas y neuronas, mejorar los resultados
- Experimentar otros inicializadores y diferentes valores de dropout
- Probar con otro dataset