Solo Borges

model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=8, input\_length=input\_seq\_len))

model.add(LSTM(32, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(16, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(32)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(32, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/10

3389/3389 [==============================] - 63s 17ms/step - loss: 7.1513 - accuracy: 0.0461 - val\_loss: 7.3686 - val\_accuracy: 0.0428

Epoch 2/10

3389/3389 [==============================] - 55s 16ms/step - loss: 6.7098 - accuracy: 0.0512 - val\_loss: 7.2523 - val\_accuracy: 0.0597

Epoch 3/10

3389/3389 [==============================] - 56s 16ms/step - loss: 6.4471 - accuracy: 0.0667 - val\_loss: 7.3989 - val\_accuracy: 0.0669

Epoch 4/10

3389/3389 [==============================] - 56s 16ms/step - loss: 6.2543 - accuracy: 0.0724 - val\_loss: 7.4813 - val\_accuracy: 0.0700

Epoch 5/10

3389/3389 [==============================] - 55s 16ms/step - loss: 6.1029 - accuracy: 0.0760 - val\_loss: 7.5791 - val\_accuracy: 0.0734

Epoch 6/10

3389/3389 [==============================] - 56s 16ms/step - loss: 5.9848 - accuracy: 0.0781 - val\_loss: 7.6732 - val\_accuracy: 0.0738

Epoch 7/10

3389/3389 [==============================] - 55s 16ms/step - loss: 5.8781 - accuracy: 0.0822 - val\_loss: 7.6834 - val\_accuracy: 0.0723

Epoch 8/10

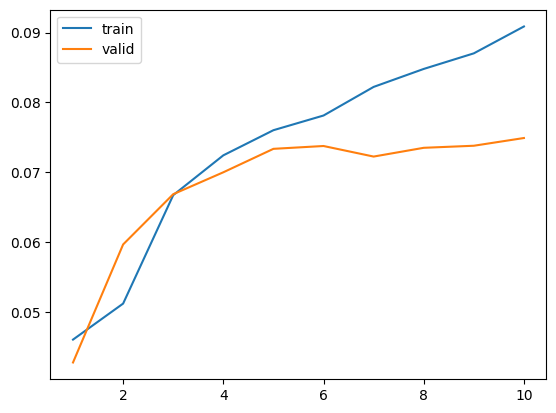
3389/3389 [==============================] - 55s 16ms/step - loss: 5.7836 - accuracy: 0.0848 - val\_loss: 7.8341 - val\_accuracy: 0.0735

Epoch 9/10

3389/3389 [==============================] - 55s 16ms/step - loss: 5.7022 - accuracy: 0.0870 - val\_loss: 7.8948 - val\_accuracy: 0.0738

Epoch 10/10

3389/3389 [==============================] - 55s 16ms/step - loss: 5.6275 - accuracy: 0.0909 - val\_loss: 7.9624 - val\_accuracy: 0.0749



Borges + Cortazar

model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=8, input\_length=input\_seq\_len))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(64, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/10

5569/5569 [==============================] - 245s 42ms/step - loss: 7.2932 - accuracy: 0.0517 - val\_loss: 7.1917 - val\_accuracy: 0.0439

Epoch 2/10

5569/5569 [==============================] - 196s 35ms/step - loss: 6.8471 - accuracy: 0.0659 - val\_loss: 7.3211 - val\_accuracy: 0.0647

Epoch 3/10

5569/5569 [==============================] - 188s 34ms/step - loss: 6.5281 - accuracy: 0.0762 - val\_loss: 7.3300 - val\_accuracy: 0.0738

Epoch 4/10

5569/5569 [==============================] - 169s 30ms/step - loss: 6.2846 - accuracy: 0.0836 - val\_loss: 7.3980 - val\_accuracy: 0.0788

Epoch 5/10

5569/5569 [==============================] - 169s 30ms/step - loss: 6.0927 - accuracy: 0.0931 - val\_loss: 7.4415 - val\_accuracy: 0.0820

Epoch 6/10

5569/5569 [==============================] - 175s 31ms/step - loss: 5.9264 - accuracy: 0.0999 - val\_loss: 7.5052 - val\_accuracy: 0.0834

Epoch 7/10

5569/5569 [==============================] - 170s 30ms/step - loss: 5.7800 - accuracy: 0.1055 - val\_loss: 7.5107 - val\_accuracy: 0.0833

Epoch 8/10

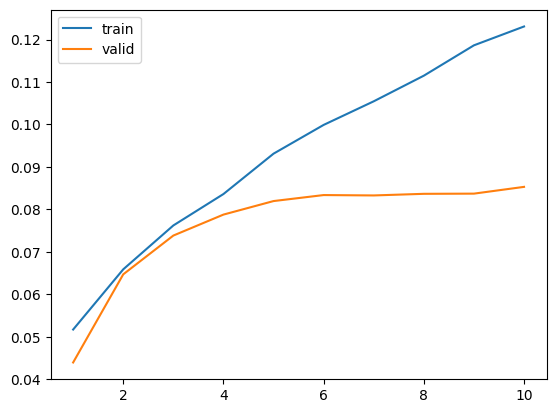
5569/5569 [==============================] - 228s 41ms/step - loss: 5.6521 - accuracy: 0.1115 - val\_loss: 7.5829 - val\_accuracy: 0.0837

Epoch 9/10

5569/5569 [==============================] - 184s 33ms/step - loss: 5.5337 - accuracy: 0.1186 - val\_loss: 7.6903 - val\_accuracy: 0.0837

Epoch 10/10

5569/5569 [==============================] - 184s 33ms/step - loss: 5.4261 - accuracy: 0.1231 - val\_loss: 7.7876 - val\_accuracy: 0.0853



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=6, input\_length=input\_seq\_len))

model.add(LSTM(100, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(100, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(100)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(100, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/10

5569/5569 [==============================] - 289s 49ms/step - loss: 7.2570 - accuracy: 0.0513 - val\_loss: 7.2351 - val\_accuracy: 0.0436

Epoch 2/10

5569/5569 [==============================] - 240s 43ms/step - loss: 6.8306 - accuracy: 0.0662 - val\_loss: 7.3012 - val\_accuracy: 0.0653

Epoch 3/10

5569/5569 [==============================] - 231s 42ms/step - loss: 6.5356 - accuracy: 0.0756 - val\_loss: 7.3322 - val\_accuracy: 0.0730

Epoch 4/10

5569/5569 [==============================] - 219s 39ms/step - loss: 6.3061 - accuracy: 0.0834 - val\_loss: 7.3885 - val\_accuracy: 0.0785

Epoch 5/10

5569/5569 [==============================] - 219s 39ms/step - loss: 6.1025 - accuracy: 0.0921 - val\_loss: 7.4449 - val\_accuracy: 0.0848

Epoch 6/10

5569/5569 [==============================] - 221s 40ms/step - loss: 5.9210 - accuracy: 0.0998 - val\_loss: 7.4681 - val\_accuracy: 0.0845

Epoch 7/10

5569/5569 [==============================] - 219s 39ms/step - loss: 5.7679 - accuracy: 0.1062 - val\_loss: 7.5355 - val\_accuracy: 0.0841

Epoch 8/10

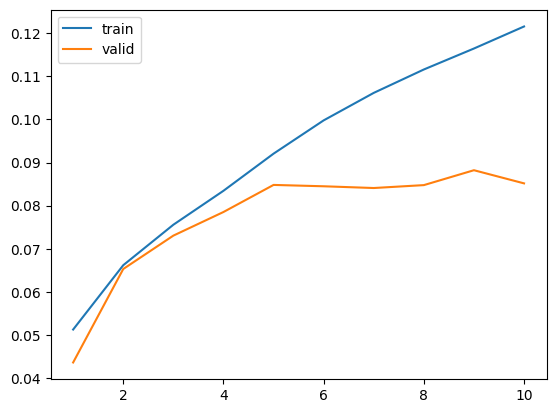
5569/5569 [==============================] - 223s 40ms/step - loss: 5.6350 - accuracy: 0.1116 - val\_loss: 7.5956 - val\_accuracy: 0.0848

Epoch 9/10

5569/5569 [==============================] - 224s 40ms/step - loss: 5.5162 - accuracy: 0.1165 - val\_loss: 7.7067 - val\_accuracy: 0.0882

Epoch 10/10

5569/5569 [==============================] - 226s 41ms/step - loss: 5.4100 - accuracy: 0.1216 - val\_loss: 7.7570 - val\_accuracy: 0.0852



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=20, input\_length=input\_seq\_len))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.1))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.1))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.1))

model.add(LSTM(64)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(64, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/10

5569/5569 [==============================] - 256s 43ms/step - loss: 7.3256 - accuracy: 0.0515 - val\_loss: 7.3476 - val\_accuracy: 0.0432

Epoch 2/10

5569/5569 [==============================] - 213s 38ms/step - loss: 6.9121 - accuracy: 0.0602 - val\_loss: 7.3502 - val\_accuracy: 0.0625

Epoch 3/10

5569/5569 [==============================] - 193s 35ms/step - loss: 6.5953 - accuracy: 0.0737 - val\_loss: 7.3855 - val\_accuracy: 0.0752

Epoch 4/10

5569/5569 [==============================] - 193s 35ms/step - loss: 6.3487 - accuracy: 0.0842 - val\_loss: 7.3935 - val\_accuracy: 0.0809

Epoch 5/10

5569/5569 [==============================] - 191s 34ms/step - loss: 6.1375 - accuracy: 0.0941 - val\_loss: 7.4360 - val\_accuracy: 0.0824

Epoch 6/10

5569/5569 [==============================] - 210s 38ms/step - loss: 5.9491 - accuracy: 0.1030 - val\_loss: 7.5520 - val\_accuracy: 0.0851

Epoch 7/10

5569/5569 [==============================] - 194s 35ms/step - loss: 5.7827 - accuracy: 0.1114 - val\_loss: 7.6146 - val\_accuracy: 0.0851

Epoch 8/10

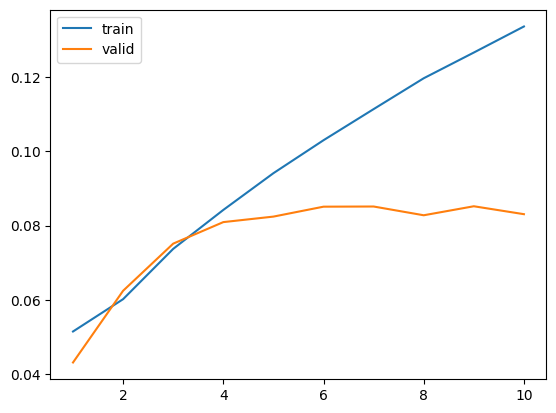
5569/5569 [==============================] - 200s 36ms/step - loss: 5.6362 - accuracy: 0.1197 - val\_loss: 7.6813 - val\_accuracy: 0.0828

Epoch 9/10

5569/5569 [==============================] - 199s 36ms/step - loss: 5.5055 - accuracy: 0.1266 - val\_loss: 7.7284 - val\_accuracy: 0.0852

Epoch 10/10

5569/5569 [==============================] - 197s 35ms/step - loss: 5.3883 - accuracy: 0.1336 - val\_loss: 7.8976 - val\_accuracy: 0.0831



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=15, input\_length=input\_seq\_len))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(128, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(200, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/10

5569/5569 [==============================] - 412s 72ms/step - loss: 7.2403 - accuracy: 0.0517 - val\_loss: 7.1660 - val\_accuracy: 0.0442

Epoch 2/10

5569/5569 [==============================] - 345s 62ms/step - loss: 6.7657 - accuracy: 0.0682 - val\_loss: 7.2582 - val\_accuracy: 0.0696

Epoch 3/10

5569/5569 [==============================] - 357s 64ms/step - loss: 6.4724 - accuracy: 0.0822 - val\_loss: 7.3300 - val\_accuracy: 0.0782

Epoch 4/10

5569/5569 [==============================] - 418s 75ms/step - loss: 6.2277 - accuracy: 0.0937 - val\_loss: 7.3751 - val\_accuracy: 0.0822

Epoch 5/10

5569/5569 [==============================] - 336s 60ms/step - loss: 6.0086 - accuracy: 0.1045 - val\_loss: 7.4340 - val\_accuracy: 0.0844

Epoch 6/10

5569/5569 [==============================] - 330s 59ms/step - loss: 5.8114 - accuracy: 0.1130 - val\_loss: 7.5211 - val\_accuracy: 0.0862

Epoch 7/10

5569/5569 [==============================] - 328s 59ms/step - loss: 5.6314 - accuracy: 0.1218 - val\_loss: 7.6048 - val\_accuracy: 0.0877

Epoch 8/10

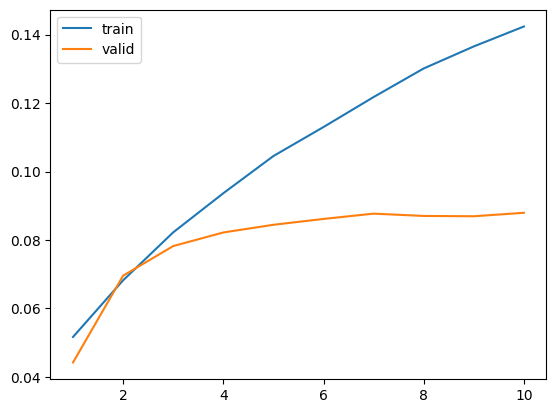
5569/5569 [==============================] - 329s 59ms/step - loss: 5.4710 - accuracy: 0.1301 - val\_loss: 7.6920 - val\_accuracy: 0.0870

Epoch 9/10

5569/5569 [==============================] - 331s 59ms/step - loss: 5.3248 - accuracy: 0.1366 - val\_loss: 7.8150 - val\_accuracy: 0.0869

Epoch 10/10

5569/5569 [==============================] - 358s 64ms/step - loss: 5.1947 - accuracy: 0.1424 - val\_loss: 7.8974 - val\_accuracy: 0.0880



NUEVO CORPUS 10K

model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=15, input\_length=input\_seq\_len))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(128, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(200, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/10

5246/5246 [==============================] - 410s 76ms/step - loss: 7.4657 - accuracy: 0.0784 - val\_loss: 7.3572 - val\_accuracy: 0.0903

Epoch 2/10

5246/5246 [==============================] - 373s 71ms/step - loss: 6.9375 - accuracy: 0.1007 - val\_loss: 7.4342 - val\_accuracy: 0.0961

Epoch 3/10

5246/5246 [==============================] - 377s 72ms/step - loss: 6.6308 - accuracy: 0.1116 - val\_loss: 7.5333 - val\_accuracy: 0.1011

Epoch 4/10

5246/5246 [==============================] - 381s 73ms/step - loss: 6.3866 - accuracy: 0.1203 - val\_loss: 7.5819 - val\_accuracy: 0.1025

Epoch 5/10

5246/5246 [==============================] - 373s 71ms/step - loss: 6.1700 - accuracy: 0.1302 - val\_loss: 7.7016 - val\_accuracy: 0.1024

Epoch 6/10

5246/5246 [==============================] - 359s 68ms/step - loss: 5.9659 - accuracy: 0.1392 - val\_loss: 7.7538 - val\_accuracy: 0.1044

Epoch 7/10

5246/5246 [==============================] - 363s 69ms/step - loss: 5.7743 - accuracy: 0.1470 - val\_loss: 7.7925 - val\_accuracy: 0.1046

Epoch 8/10

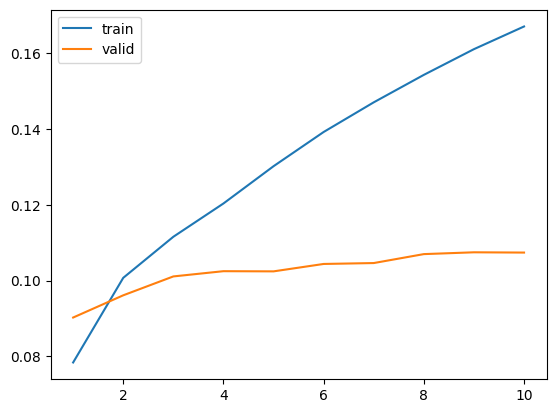
5246/5246 [==============================] - 368s 70ms/step - loss: 5.6016 - accuracy: 0.1543 - val\_loss: 8.0189 - val\_accuracy: 0.1070

Epoch 9/10

5246/5246 [==============================] - 359s 68ms/step - loss: 5.4435 - accuracy: 0.1610 - val\_loss: 8.0406 - val\_accuracy: 0.1075

Epoch 10/10

5246/5246 [==============================] - 359s 68ms/step - loss: 5.2950 - accuracy: 0.1670 - val\_loss: 8.3557 - val\_accuracy: 0.1074



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=10, input\_length=input\_seq\_len))

model.add(LSTM(32, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(32)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(64, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

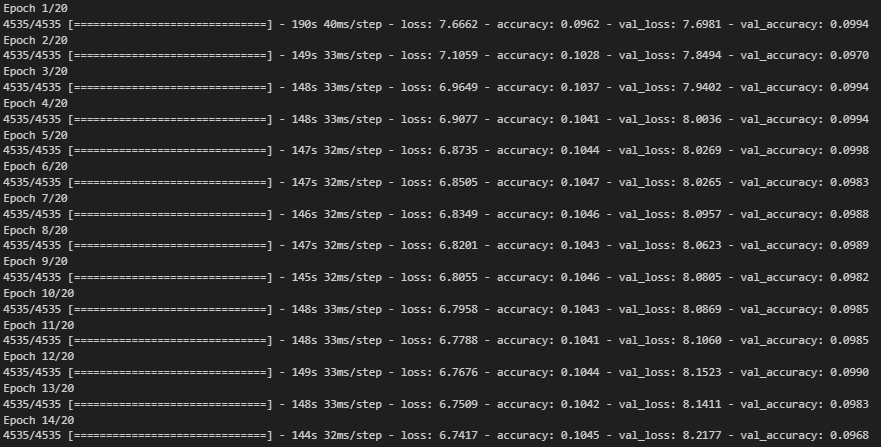
model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

opt = keras.optimizers.Adam(learning\_rate=0.02)

model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])

model.summary()



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=15, input\_length=input\_seq\_len))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(64)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(64, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

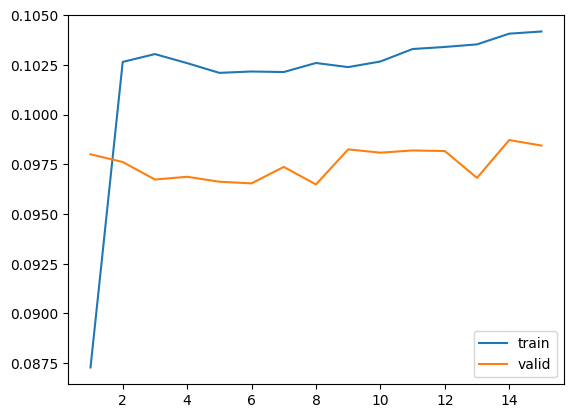
model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

opt = keras.optimizers.Adam(learning\_rate=0.02)

model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])

model.summary()



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=50, input\_length=input\_seq\_len))

# model.add(LSTM(128, return\_sequences=True))

# model.add(Dropout(0.3))

model.add(LSTM(256)) # La última capa LSTM no lleva return\_sequences

# model.add(Dense(64, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer="adam", metrics=['accuracy'])

model.summary()

Epoch 1/15

4535/4535 [==============================] - 308s 66ms/step - loss: 7.4932 - accuracy: 0.0898 - val\_loss: 7.3186 - val\_accuracy: 0.1016

Epoch 2/15

4535/4535 [==============================] - 233s 51ms/step - loss: 6.6899 - accuracy: 0.1180 - val\_loss: 7.2332 - val\_accuracy: 0.1202

Epoch 3/15

4535/4535 [==============================] - 232s 51ms/step - loss: 6.2085 - accuracy: 0.1412 - val\_loss: 7.3137 - val\_accuracy: 0.1252

Epoch 4/15

4535/4535 [==============================] - 231s 51ms/step - loss: 5.8024 - accuracy: 0.1596 - val\_loss: 7.4403 - val\_accuracy: 0.1274

Epoch 5/15

4535/4535 [==============================] - 231s 51ms/step - loss: 5.3842 - accuracy: 0.1821 - val\_loss: 7.6090 - val\_accuracy: 0.1242

Epoch 6/15

4535/4535 [==============================] - 231s 51ms/step - loss: 4.9627 - accuracy: 0.2060 - val\_loss: 7.8248 - val\_accuracy: 0.1211

Epoch 7/15

4535/4535 [==============================] - 230s 51ms/step - loss: 4.5408 - accuracy: 0.2404 - val\_loss: 8.0609 - val\_accuracy: 0.1149

Epoch 8/15

4535/4535 [==============================] - 231s 51ms/step - loss: 4.1353 - accuracy: 0.2824 - val\_loss: 8.2954 - val\_accuracy: 0.1130

Epoch 9/15

4535/4535 [==============================] - 236s 52ms/step - loss: 3.7520 - accuracy: 0.3309 - val\_loss: 8.4725 - val\_accuracy: 0.1073

Epoch 10/15

4535/4535 [==============================] - 237s 52ms/step - loss: 3.3963 - accuracy: 0.3825 - val\_loss: 8.6620 - val\_accuracy: 0.1038

Epoch 11/15

4535/4535 [==============================] - 229s 51ms/step - loss: 3.0689 - accuracy: 0.4327 - val\_loss: 8.8423 - val\_accuracy: 0.1005

Epoch 12/15

4535/4535 [==============================] - 233s 51ms/step - loss: 2.7762 - accuracy: 0.4790 - val\_loss: 9.0084 - val\_accuracy: 0.0994

Epoch 13/15

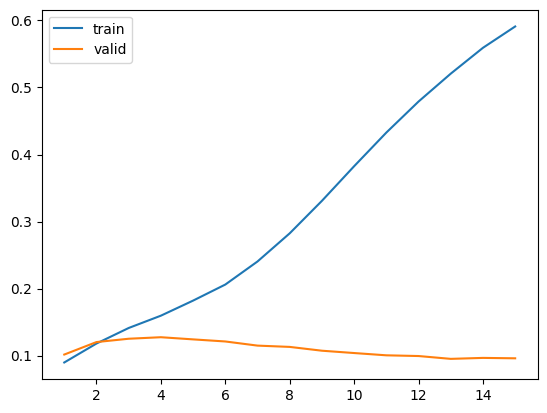
4535/4535 [==============================] - 232s 51ms/step - loss: 2.5168 - accuracy: 0.5206 - val\_loss: 9.1819 - val\_accuracy: 0.0952

Epoch 14/15

4535/4535 [==============================] - 235s 52ms/step - loss: 2.2920 - accuracy: 0.5592 - val\_loss: 9.3296 - val\_accuracy: 0.0966

Epoch 15/15

4535/4535 [==============================] - 242s 53ms/step - loss: 2.0934 - accuracy: 0.5910 - val\_loss: 9.4551 - val\_accuracy: 0.0960



model = Sequential()

# Embedding:

# input\_seq\_len = 3 --> ingreso 3 palabras

# input\_dim = vocab\_size --> 15002 palabras distintas

# output\_dim = 5 --> crear embeddings de tamaño 3 (tamaño variable y ajustable)

model.add(Embedding(input\_dim=vocab\_size+1, output\_dim=20, input\_length=input\_seq\_len))

# model.add(LSTM(128, return\_sequences=True))

# model.add(Dropout(0.3))

model.add(LSTM(128)) # La última capa LSTM no lleva return\_sequences

model.add(Dense(128, activation='relu'))

# Predicción de clasificación con softmax

# La salida vuelve al espacio de 15002 palabras posibles

model.add(Dense(vocab\_size, activation='softmax'))

# Clasificación multiple categórica --> loss = categorical\_crossentropy

model.compile(loss='categorical\_crossentropy', optimizer="adam", metrics=['accuracy'])

model.summary()

Epoch 1/15

4535/4535 [==============================] - 270s 58ms/step - loss: 7.4148 - accuracy: 0.0820 - val\_loss: 7.3727 - val\_accuracy: 0.0989

Epoch 2/15

4535/4535 [==============================] - 213s 47ms/step - loss: 6.8473 - accuracy: 0.1090 - val\_loss: 7.3933 - val\_accuracy: 0.1081

Epoch 3/15

4535/4535 [==============================] - 211s 47ms/step - loss: 6.4395 - accuracy: 0.1249 - val\_loss: 7.4890 - val\_accuracy: 0.1158

Epoch 4/15

4535/4535 [==============================] - 210s 46ms/step - loss: 6.0999 - accuracy: 0.1440 - val\_loss: 7.5654 - val\_accuracy: 0.1216

Epoch 5/15

4535/4535 [==============================] - 209s 46ms/step - loss: 5.7634 - accuracy: 0.1627 - val\_loss: 7.6880 - val\_accuracy: 0.1227

Epoch 6/15

4535/4535 [==============================] - 214s 47ms/step - loss: 5.4695 - accuracy: 0.1805 - val\_loss: 7.9331 - val\_accuracy: 0.1251

Epoch 7/15

4535/4535 [==============================] - 211s 46ms/step - loss: 5.1943 - accuracy: 0.1963 - val\_loss: 8.1967 - val\_accuracy: 0.1221

Epoch 8/15

4535/4535 [==============================] - 215s 47ms/step - loss: 4.9051 - accuracy: 0.2110 - val\_loss: 8.4979 - val\_accuracy: 0.1219

Epoch 9/15

4535/4535 [==============================] - 214s 47ms/step - loss: 4.5999 - accuracy: 0.2251 - val\_loss: 9.0080 - val\_accuracy: 0.1197

Epoch 10/15

4535/4535 [==============================] - 217s 48ms/step - loss: 4.2745 - accuracy: 0.2419 - val\_loss: 9.5293 - val\_accuracy: 0.1145

Epoch 11/15

4535/4535 [==============================] - 225s 50ms/step - loss: 3.9542 - accuracy: 0.2660 - val\_loss: 10.2355 - val\_accuracy: 0.1093

Epoch 12/15

4535/4535 [==============================] - 219s 48ms/step - loss: 3.6603 - accuracy: 0.2953 - val\_loss: 10.8713 - val\_accuracy: 0.1012

Epoch 13/15

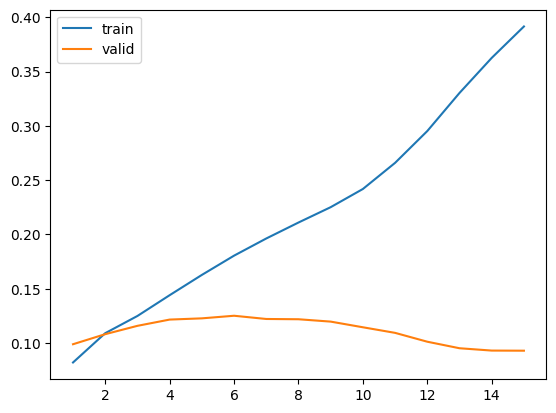
4535/4535 [==============================] - 223s 49ms/step - loss: 3.3973 - accuracy: 0.3304 - val\_loss: 11.5309 - val\_accuracy: 0.0952

Epoch 14/15

4535/4535 [==============================] - 226s 50ms/step - loss: 3.1735 - accuracy: 0.3627 - val\_loss: 12.2548 - val\_accuracy: 0.0930

Epoch 15/15

4535/4535 [==============================] - 225s 50ms/step - loss: 2.9812 - accuracy: 0.3916 - val\_loss: 12.6682 - val\_accuracy: 0.0929



# Crear modelo con Keras optimizado para un vocabulario de 25000 palabras y 222391 secuencias de entrada

model = Sequential()

model.add(Embedding(input\_dim=len(model\_w2v.wv.index\_to\_key), output\_dim=100, input\_length=input\_seq\_len, weights=[model\_w2v.wv.vectors], trainable=False))

model.add(GRU(256, return\_sequences=True))

model.add(GRU(128))

model.add(Dropout(0.2))

model.add(Dense(150, activation='relu'))

model.add(Dense(output\_size, activation='softmax'))

# Compilar el modelo

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

Epoch 1/20

1390/1390 [==============================] - 182s 123ms/step - loss: 7.2188 - accuracy: 0.0510 - val\_loss: 7.0308 - val\_accuracy: 0.0472

Epoch 2/20

1390/1390 [==============================] - 116s 83ms/step - loss: 6.7876 - accuracy: 0.0715 - val\_loss: 6.9126 - val\_accuracy: 0.0728

Epoch 3/20

1390/1390 [==============================] - 117s 84ms/step - loss: 6.5444 - accuracy: 0.0821 - val\_loss: 6.8736 - val\_accuracy: 0.0829

Epoch 4/20

1390/1390 [==============================] - 113s 81ms/step - loss: 6.3504 - accuracy: 0.0902 - val\_loss: 6.8271 - val\_accuracy: 0.0944

Epoch 5/20

1390/1390 [==============================] - 112s 80ms/step - loss: 6.1747 - accuracy: 0.0979 - val\_loss: 6.8708 - val\_accuracy: 0.1027

Epoch 6/20

1390/1390 [==============================] - 112s 80ms/step - loss: 6.0243 - accuracy: 0.1045 - val\_loss: 6.8471 - val\_accuracy: 0.1061

Epoch 7/20

1390/1390 [==============================] - 113s 81ms/step - loss: 5.8897 - accuracy: 0.1088 - val\_loss: 6.9031 - val\_accuracy: 0.1119

Epoch 8/20

1390/1390 [==============================] - 111s 80ms/step - loss: 5.7663 - accuracy: 0.1129 - val\_loss: 6.9735 - val\_accuracy: 0.1150

Epoch 9/20

1390/1390 [==============================] - 112s 81ms/step - loss: 5.6558 - accuracy: 0.1172 - val\_loss: 7.0160 - val\_accuracy: 0.1194

Epoch 10/20

1390/1390 [==============================] - 112s 80ms/step - loss: 5.5493 - accuracy: 0.1204 - val\_loss: 7.0643 - val\_accuracy: 0.1211

Epoch 11/20

1390/1390 [==============================] - 112s 80ms/step - loss: 5.4501 - accuracy: 0.1240 - val\_loss: 7.1550 - val\_accuracy: 0.1232

Epoch 12/20

1390/1390 [==============================] - 112s 80ms/step - loss: 5.3582 - accuracy: 0.1265 - val\_loss: 7.2085 - val\_accuracy: 0.1256

Epoch 13/20

1390/1390 [==============================] - 112s 81ms/step - loss: 5.2695 - accuracy: 0.1288 - val\_loss: 7.3251 - val\_accuracy: 0.1294

Epoch 14/20

1390/1390 [==============================] - 112s 80ms/step - loss: 5.1859 - accuracy: 0.1319 - val\_loss: 7.4414 - val\_accuracy: 0.1304

Epoch 15/20

1390/1390 [==============================] - 112s 80ms/step - loss: 5.1053 - accuracy: 0.1340 - val\_loss: 7.6076 - val\_accuracy: 0.1302

Epoch 16/20

1390/1390 [==============================] - 110s 79ms/step - loss: 5.0282 - accuracy: 0.1350 - val\_loss: 7.7218 - val\_accuracy: 0.1317

Epoch 17/20

1390/1390 [==============================] - 110s 79ms/step - loss: 4.9510 - accuracy: 0.1384 - val\_loss: 7.9254 - val\_accuracy: 0.1344

Epoch 18/20

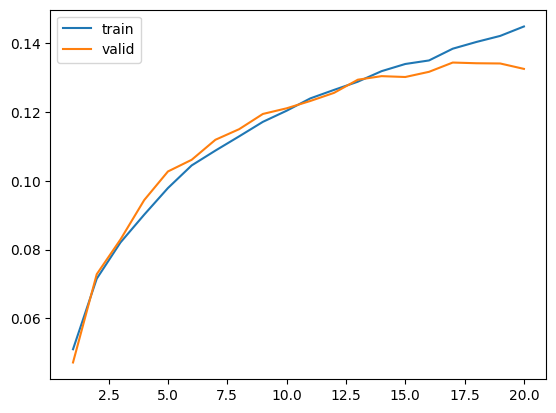
1390/1390 [==============================] - 111s 80ms/step - loss: 4.8761 - accuracy: 0.1404 - val\_loss: 8.0589 - val\_accuracy: 0.1342

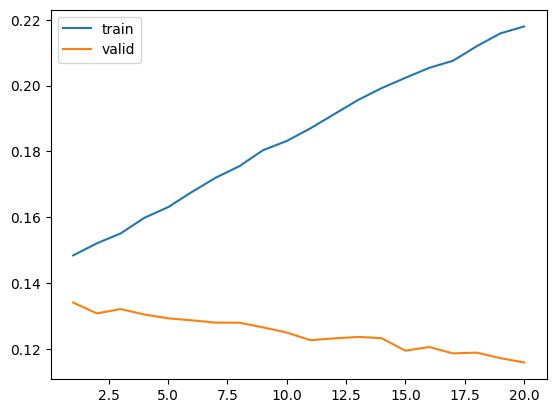
Epoch 19/20

1390/1390 [==============================] - 110s 79ms/step - loss: 4.8032 - accuracy: 0.1421 - val\_loss: 8.2726 - val\_accuracy: 0.1341

Epoch 20/20

1390/1390 [==============================] - 110s 79ms/step - loss: 4.7309 - accuracy: 0.1449 - val\_loss: 8.3895 - val\_accuracy: 0.1326





# Crear modelo con Keras optimizado para un vocabulario de 25000 palabras y 222391 secuencias de entrada

model = Sequential()

model.add(Embedding(vocab\_size+1, 100, input\_length=input\_seq\_len, trainable=True))

model.add(GRU(128, return\_sequences=True))

model.add(Dropout(0.5))

model.add(GRU(128, return\_sequences=True))

model.add(Dropout(0.5))

model.add(GRU(128))

model.add(Dense(128, activation='relu'))

model.add(Dense(output\_size, activation='softmax'))

# Compilar el modelo

model.compile(loss='categorical\_crossentropy', optimizer='adam' , metrics=['accuracy'])

model.summary()

Epoch 1/20

97/97 [==============================] - 143s 1s/step - loss: 7.9680 - accuracy: 0.0484 - val\_loss: 7.3917 - val\_accuracy: 0.0440

Epoch 2/20

97/97 [==============================] - 109s 1s/step - loss: 7.1411 - accuracy: 0.0512 - val\_loss: 7.4912 - val\_accuracy: 0.0440

Epoch 3/20

97/97 [==============================] - 106s 1s/step - loss: 7.1087 - accuracy: 0.0512 - val\_loss: 7.5269 - val\_accuracy: 0.0440

Epoch 4/20

97/97 [==============================] - 103s 1s/step - loss: 7.0855 - accuracy: 0.0496 - val\_loss: 7.6023 - val\_accuracy: 0.0440

Epoch 5/20

97/97 [==============================] - 108s 1s/step - loss: 7.0695 - accuracy: 0.0511 - val\_loss: 7.5938 - val\_accuracy: 0.0438

Epoch 6/20

97/97 [==============================] - 103s 1s/step - loss: 7.0274 - accuracy: 0.0511 - val\_loss: 7.4959 - val\_accuracy: 0.0466

Epoch 7/20

97/97 [==============================] - 100s 1s/step - loss: 6.8578 - accuracy: 0.0641 - val\_loss: 7.3135 - val\_accuracy: 0.0611

Epoch 8/20

97/97 [==============================] - 110s 1s/step - loss: 6.6379 - accuracy: 0.0722 - val\_loss: 7.2986 - val\_accuracy: 0.0620

Epoch 9/20

97/97 [==============================] - 107s 1s/step - loss: 6.5051 - accuracy: 0.0763 - val\_loss: 7.2747 - val\_accuracy: 0.0664

Epoch 10/20

97/97 [==============================] - 109s 1s/step - loss: 6.3880 - accuracy: 0.0822 - val\_loss: 7.3061 - val\_accuracy: 0.0710

Epoch 11/20

97/97 [==============================] - 105s 1s/step - loss: 6.2654 - accuracy: 0.0880 - val\_loss: 7.3434 - val\_accuracy: 0.0706

Epoch 12/20

97/97 [==============================] - 107s 1s/step - loss: 6.1402 - accuracy: 0.0926 - val\_loss: 7.3691 - val\_accuracy: 0.0719

Epoch 13/20

97/97 [==============================] - 108s 1s/step - loss: 6.0137 - accuracy: 0.0972 - val\_loss: 7.4233 - val\_accuracy: 0.0733

Epoch 14/20

97/97 [==============================] - 98s 1s/step - loss: 5.8872 - accuracy: 0.1027 - val\_loss: 7.4665 - val\_accuracy: 0.0751

Epoch 15/20

97/97 [==============================] - 93s 958ms/step - loss: 5.7690 - accuracy: 0.1070 - val\_loss: 7.5148 - val\_accuracy: 0.0779

Epoch 16/20

97/97 [==============================] - 101s 1s/step - loss: 5.6630 - accuracy: 0.1114 - val\_loss: 7.6027 - val\_accuracy: 0.0764

Epoch 17/20

97/97 [==============================] - 101s 1s/step - loss: 5.5533 - accuracy: 0.1171 - val\_loss: 7.7678 - val\_accuracy: 0.0808

Epoch 18/20

97/97 [==============================] - 96s 993ms/step - loss: 5.4540 - accuracy: 0.1210 - val\_loss: 7.8284 - val\_accuracy: 0.0798

Epoch 19/20

97/97 [==============================] - 102s 1s/step - loss: 5.3626 - accuracy: 0.1264 - val\_loss: 7.9551 - val\_accuracy: 0.0813

Epoch 20/20

97/97 [==============================] - 100s 1s/step - loss: 5.2774 - accuracy: 0.1301 - val\_loss: 8.0900 - val\_accuracy: 0.0828

