from google.colab import drive

# monta tu unidad de google drive

drive.mount('/content/drive')

# cambiamos el directorio de trabajo a la carpeta donde está almacenado el dataset

import os

os.chdir('/content/drive/My Drive/Colab Notebooks/HomoMex\_2024/public\_data\_dev\_phase')

tomar los archivos de dataset de traing y dataset

import pandas as pd

df\_train = pd.read\_csv("train.csv",  sep=',')

df\_test = pd.read\_csv("val.csv",  sep=',')

print('tamaño training',df\_train.shape)

#print('tamaño test',df\_test.shape)

print('tamaño test',df\_test.shape)

df\_train.head()

#df\_val.head()

Tamaño para cada uno de los archivos

df\_test.describe()

counts\_train = df\_train['label'].value\_counts().to\_dict()

print("Distribución en training:", counts\_train)

counts\_test = df\_test['label'].value\_counts().to\_dict()

print("Distribución en test:", counts\_test)

Distribución de clases

import matplotlib.pyplot as plt

fig, (ax\_train, ax\_test) = plt.subplots(1, 2, figsize=(20, 10))

fig.suptitle('Distribución de clases')

ax\_train.bar(list(counts\_train.keys()), counts\_train.values(), width=0.4)

ax\_train.set\_xticklabels(list(counts\_train.keys()), rotation=45)

ax\_train.set\_title('Conjunto de entrenamiento')

ax\_train.set\_xlabel('Clases')

ax\_train.set\_ylabel('Número de Instancias')

ax\_test.bar(list(counts\_test.keys()), counts\_test.values(), width=0.4)

ax\_test.set\_xticklabels(list(counts\_test.keys()), rotation=45)

ax\_test.set\_title('Conjunto de evaluación')

ax\_test.set\_xlabel('Clases')

ax\_test.set\_ylabel('Número de Instancias')

Graficamos los elementos de la clase del label

Label Codign

Label encoding esta tarea consiste en trasformar las labels en formato a texto a un formato numérico

from sklearn.preprocessing import LabelEncoder

from keras.utils import to\_categorical

y\_train = df\_train['label'].tolist()

y\_test = df\_test['label'].tolist()

le = LabelEncoder()

print("antes de transform: ", y\_train[:10])

y\_train = le.fit\_transform(y\_train)

print("después de transform:", y\_train[:10])

y\_train\_cat = to\_categorical(y\_train)

print("después de transform:", y\_train\_cat[:10])

y\_test = le.fit\_transform(y\_test)

y\_test\_cat = to\_categorical(y\_test)

# guardamos el conjunto de labels y el número

LABELS = le.classes\_

NUM\_LABELS = len(LABELS)

idx2label={}

label2idx={}

for index, label in enumerate(LABELS):

    label2idx.update([(label, index)])

    idx2label.update([(index, label)])

# print(idx2label)

print('Labels:', label2idx)

print()

from keras.preprocessing.text import Tokenizer

X\_train = df\_train['content'].tolist()

tokenizer = Tokenizer(oov\_token = True)

tokenizer.fit\_on\_texts(X\_train)

tokenizer.word\_index['<PAD>'] = 0

NUM\_WORDS=len(tokenizer.word\_index)

print("Tamaño del vocabulario ={}".format(NUM\_WORDS))

capas neuronales de CNN y BLTS

from keras.models import Sequential

from keras.layers import Flatten, Dense, Embedding, Conv1D, MaxPooling1D

model = Sequential()

EMBEDDING\_SIZE=300 #probar con 200, 300

model.add(Embedding(NUM\_WORDS, EMBEDDING\_SIZE, input\_length=MAX\_LEN))

model.add(Conv1D(filters=128, kernel\_size=4, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

model.add(Conv1D(filters=64, kernel\_size=3, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

model.add(Conv1D(filters=32, kernel\_size=2, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

from keras.layers import Bidirectional, LSTM, Dropout, Dense

DROPOUT = 0.4

model.add(Bidirectional(LSTM(60, return\_sequences=True, recurrent\_dropout=0.2)))

# Añadimos una capaz droput después de la capa bilstm

model.add(Dropout(DROPOUT))

model.add(Bidirectional(LSTM(32, recurrent\_dropout=0.2)))

model.add(Dropout(DROPOUT))

model.add(Dense(60, activation='relu'))

model.add(Flatten())

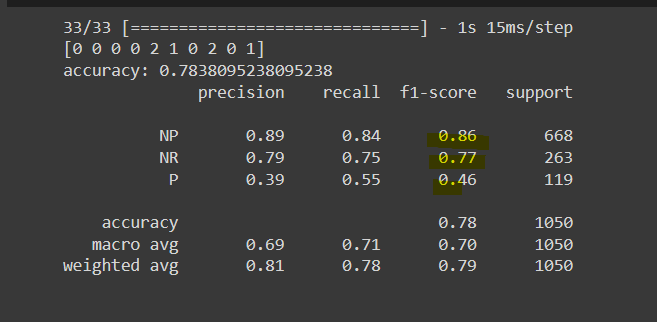
model.add(Dense(50, activation='relu'))

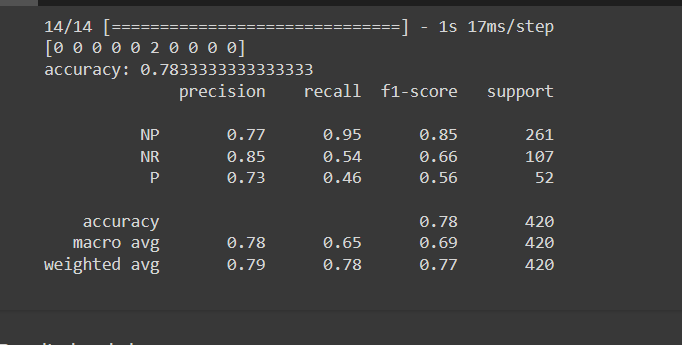
model.add(Dense(NUM\_LABELS, activation='softmax'))

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

model.summary()

resultados del modelo pre entrenado en basa a los valores



15 de abril del 2024

Capas de la neurona

from keras.models import Sequential

from keras.layers import Flatten, Dense, Embedding, Conv1D, MaxPooling1D

# campa de CNN

model = Sequential()

EMBEDDING\_SIZE=300 #probar con 200, 300

model.add(Embedding(NUM\_WORDS, EMBEDDING\_SIZE, input\_length=MAX\_LEN))

model.add(Conv1D(filters=128, kernel\_size=4, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

model.add(Conv1D(filters=64, kernel\_size=3, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

model.add(Conv1D(filters=32, kernel\_size=2, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

# capa de bilstm

from keras.layers import Bidirectional, LSTM, Dropout, Dense

DROPOUT = 0.4

model.add(Bidirectional(LSTM(60, return\_sequences=True, recurrent\_dropout=0.2)))

# Añadimos una capaz droput después de la capa bilstm

model.add(Dropout(DROPOUT))

model.add(Bidirectional(LSTM(32, recurrent\_dropout=0.2)))

model.add(Dropout(DROPOUT))

model.add(Dense(10, activation='relu'))

sofmax realizado

model.add(Flatten())

model.add(Dense(50, activation='relu'))

model.add(Dense(NUM\_LABELS, activation='softmax'))

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

model.summary()

