{

"cells": [

{

"cell\_type": "code",

"execution\_count": 12,

"id": "bbe0b8f7",

"metadata": {},

"outputs": [],

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.preprocessing import LabelEncoder\n",

"from keras.models import Model\n",

"from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding\n",

"from keras.optimizers import RMSprop\n",

"from keras.preprocessing.text import Tokenizer\n",

"from keras\_preprocessing import sequence\n",

"from keras.utils import to\_categorical\n",

"from keras.callbacks import EarlyStopping\n",

"from keras.models import load\_model\n",

"%matplotlib inline"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2e9ce340",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": 14,

"id": "78a9b584",

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>v1</th>\n",

" <th>v2</th>\n",

" <th>Unnamed: 2</th>\n",

" <th>Unnamed: 3</th>\n",

" <th>Unnamed: 4</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>ham</td>\n",

" <td>Go until jurong point, crazy.. Available only ...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>ham</td>\n",

" <td>Ok lar... Joking wif u oni...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>spam</td>\n",

" <td>Free entry in 2 a wkly comp to win FA Cup fina...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>ham</td>\n",

" <td>U dun say so early hor... U c already then say...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>ham</td>\n",

" <td>Nah I don't think he goes to usf, he lives aro...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" v1 v2 Unnamed: 2 \\\n",

"0 ham Go until jurong point, crazy.. Available only ... NaN \n",

"1 ham Ok lar... Joking wif u oni... NaN \n",

"2 spam Free entry in 2 a wkly comp to win FA Cup fina... NaN \n",

"3 ham U dun say so early hor... U c already then say... NaN \n",

"4 ham Nah I don't think he goes to usf, he lives aro... NaN \n",

"\n",

" Unnamed: 3 Unnamed: 4 \n",

"0 NaN NaN \n",

"1 NaN NaN \n",

"2 NaN NaN \n",

"3 NaN NaN \n",

"4 NaN NaN "

]

},

"execution\_count": 14,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df = pd.read\_csv('spam.csv',delimiter=',',encoding='latin-1')\n",

"df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 15,

"id": "e8514131",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"<class 'pandas.core.frame.DataFrame'>\n",

"RangeIndex: 5572 entries, 0 to 5571\n",

"Data columns (total 2 columns):\n",

" # Column Non-Null Count Dtype \n",

"--- ------ -------------- ----- \n",

" 0 v1 5572 non-null object\n",

" 1 v2 5572 non-null object\n",

"dtypes: object(2)\n",

"memory usage: 87.2+ KB\n"

]

}

],

"source": [

"df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)\n",

"df.info()"

]

},

{

"cell\_type": "code",

"execution\_count": 16,

"id": "8783fcef",

"metadata": {},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"C:\\Users\\sathi\\anaconda3\\lib\\site-packages\\seaborn\\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.\n",

" warnings.warn(\n"

]

},

{

"data": {

"text/plain": [

"Text(0.5, 1.0, 'Number of ham and spam messages')"

]

},

"execution\_count": 16,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"# data distribution\n",

"sns.countplot(df.v1)\n",

"plt.xlabel('Label')\n",

"plt.title('Number of ham and spam messages')"

]

},

{

"cell\_type": "code",

"execution\_count": 17,

"id": "93f91193",

"metadata": {},

"outputs": [],

"source": [

"# splitting data into input and output\n",

"X = df.v2\n",

"Y = df.v1\n",

"le = LabelEncoder()\n",

"Y = le.fit\_transform(Y)\n",

"Y = Y.reshape(-1,1)"

]

},

{

"cell\_type": "code",

"execution\_count": 18,

"id": "afe2d75d",

"metadata": {},

"outputs": [],

"source": [

"# test and train split\n",

"X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.15)"

]

},

{

"cell\_type": "code",

"execution\_count": 19,

"id": "4f7807e5",

"metadata": {},

"outputs": [],

"source": [

"max\_words = 1000\n",

"max\_len = 150\n",

"tok = Tokenizer(num\_words=max\_words)\n",

"tok.fit\_on\_texts(X\_train)\n",

"sequences = tok.texts\_to\_sequences(X\_train)\n",

"\n",

"sequences\_matrix = sequence.pad\_sequences(sequences,maxlen=max\_len)\n"

]

},

{

"cell\_type": "code",

"execution\_count": 20,

"id": "304aa1d6",

"metadata": {},

"outputs": [],

"source": [

"#layers of the model\n",

"inputs = Input(name='inputs',shape=[max\_len])\n",

"layer = Embedding(max\_words,50,input\_length=max\_len)(inputs)\n",

"layer = LSTM(64)(layer)\n",

"layer = Dense(256,name='FC1')(layer)\n",

"layer = Activation('relu')(layer)\n",

"layer = Dropout(0.5)(layer)\n",

"layer = Dense(1,name='out\_layer')(layer)\n",

"layer = Activation('sigmoid')(layer)"

]

},

{

"cell\_type": "code",

"execution\_count": 21,

"id": "44edc1b2",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Model: \"model\"\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

" Layer (type) Output Shape Param # \n",

"=================================================================\n",

" inputs (InputLayer) [(None, 150)] 0 \n",

" \n",

" embedding (Embedding) (None, 150, 50) 50000 \n",

" \n",

" lstm (LSTM) (None, 64) 29440 \n",

" \n",

" FC1 (Dense) (None, 256) 16640 \n",

" \n",

" activation (Activation) (None, 256) 0 \n",

" \n",

" dropout (Dropout) (None, 256) 0 \n",

" \n",

" out\_layer (Dense) (None, 1) 257 \n",

" \n",

" activation\_1 (Activation) (None, 1) 0 \n",

" \n",

"=================================================================\n",

"Total params: 96,337\n",

"Trainable params: 96,337\n",

"Non-trainable params: 0\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

]

}

],

"source": [

"model = Model(inputs=inputs,outputs=layer)\n",

"model.summary()\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 22,

"id": "57be216e",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Epoch 1/10\n",

"30/30 [==============================] - 14s 282ms/step - loss: 0.3317 - accuracy: 0.8720 - val\_loss: 0.1777 - val\_accuracy: 0.9768\n",

"Epoch 2/10\n",

"30/30 [==============================] - 7s 226ms/step - loss: 0.0930 - accuracy: 0.9776 - val\_loss: 0.0651 - val\_accuracy: 0.9842\n"

]

},

{

"data": {

"text/plain": [

"<keras.callbacks.History at 0x222d39d8d00>"

]

},

"execution\_count": 22,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,\n",

" validation\_split=0.2,callbacks=[EarlyStopping(monitor='val\_loss',min\_delta=0.0001)])"

]

},

{

"cell\_type": "code",

"execution\_count": 30,

"id": "ce5258f0",

"metadata": {},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"WARNING:absl:Found untraced functions such as lstm\_cell\_layer\_call\_fn, lstm\_cell\_layer\_call\_and\_return\_conditional\_losses while saving (showing 2 of 2). These functions will not be directly callable after loading.\n"

]

},

{

"name": "stdout",

"output\_type": "stream",

"text": [

"INFO:tensorflow:Assets written to: my\_model\\assets\n"

]

},

{

"name": "stderr",

"output\_type": "stream",

"text": [

"INFO:tensorflow:Assets written to: my\_model\\assets\n"

]

}

],

"source": [

"# saving a model\n",

"model.save(\"my\_model\")"

]

},

{

"cell\_type": "markdown",

"id": "3e3b1525",

"metadata": {},

"source": [

"# loading model\n",

"# model = load\_model('saved\_model/my\_model')"

]

},

{

"cell\_type": "code",

"execution\_count": 31,

"id": "cda2a0ca",

"metadata": {},

"outputs": [],

"source": [

"test\_sequences = tok.texts\_to\_sequences(X\_test)\n",

"test\_sequences\_matrix = sequence.pad\_sequences(test\_sequences,maxlen=max\_len)"

]

},

{

"cell\_type": "code",

"execution\_count": 32,

"id": "38d64f40",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"27/27 [==============================] - 1s 34ms/step - loss: 0.0686 - accuracy: 0.9844\n"

]

}

],

"source": [

"accr = model.evaluate(test\_sequences\_matrix,Y\_test)"

]

},

{

"cell\_type": "code",

"execution\_count": 33,

"id": "26ad9585",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Test set\n",

" Loss: 0.069\n",

" Accuracy: 0.984\n"

]

}

],

"source": [

"print('Test set\\n Loss: {:0.3f}\\n Accuracy: {:0.3f}'.format(accr[0],accr[1]))"

]

},

{

"cell\_type": "code",

"execution\_count": 34,

"id": "cfe48015",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"27/27 [==============================] - 1s 33ms/step\n",

"[[0.0052104 ]\n",

" [0.00270706]\n",

" [0.01115318]\n",

" [0.00448523]\n",

" [0.02838335]\n",

" [0.00183202]\n",

" [0.43712318]\n",

" [0.00751833]\n",

" [0.00228402]\n",

" [0.78971624]]\n"

]

}

],

"source": [

"y\_pred = model.predict(test\_sequences\_matrix)\n",

"print(y\_pred[0:10])"

]

},

{

"cell\_type": "code",

"execution\_count": 35,

"id": "f0154ae1",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[[0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [1]]\n"

]

}

],

"source": [

"print(Y\_test[0:10])"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "a66043e8",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "ce1e195e",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.12"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}