{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "code",

"source": [

"from google.colab import drive\n",

"drive.mount('/content/drive')"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "AHMosaYVIXLe",

"outputId": "609e3fe8-92c2-4ab1-bf29-0d324ab275c0"

},

"execution\_count": 3,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Mounted at /content/drive\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"## Dataset downloaded\n",

"\n",

"\n",

"## Import required library"

],

"metadata": {

"id": "2\_fSBtodE0oH"

}

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {

"id": "HPet1T88Ey2q"

},

"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",

"%matplotlib inline"

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Read dataset and do pre-processing\*\*"

],

"metadata": {

"id": "p9bzkL1eIxHt"

}

},

{

"cell\_type": "code",

"source": [

"df = pd.read\_csv('/content/drive/MyDrive/units/spam.csv',delimiter=',',encoding='latin-1')"

],

"metadata": {

"id": "MeNeGCVmImuA"

},

"execution\_count": 5,

"outputs": []

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "XGxCdyYsI5Lu"

},

"execution\_count": 6,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df.head()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 206

},

"id": "7524X8ptI8SF",

"outputId": "ff3da6f2-7377-44e4-e872-96a6dba687b0"

},

"execution\_count": 7,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

" v1 v2\n",

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

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

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

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

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

],

"text/html": [

"\n",

" <div id=\"df-88228d97-4393-42c3-a3a5-47a8d1a95afe\">\n",

" <div class=\"colab-df-container\">\n",

" <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",

" </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",

" </tr>\n",

" <tr>\n",

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

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

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

" </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",

" </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",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>\n",

" <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-88228d97-4393-42c3-a3a5-47a8d1a95afe')\"\n",

" title=\"Convert this dataframe to an interactive table.\"\n",

" style=\"display:none;\">\n",

" \n",

" <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",

" width=\"24px\">\n",

" <path d=\"M0 0h24v24H0V0z\" fill=\"none\"/>\n",

" <path d=\"M18.56 5.44l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94zm-11 1L8.5 8.5l.94-2.06 2.06-.94-2.06-.94L8.5 2.5l-.94 2.06-2.06.94zm10 10l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94z\"/><path d=\"M17.41 7.96l-1.37-1.37c-.4-.4-.92-.59-1.43-.59-.52 0-1.04.2-1.43.59L10.3 9.45l-7.72 7.72c-.78.78-.78 2.05 0 2.83L4 21.41c.39.39.9.59 1.41.59.51 0 1.02-.2 1.41-.59l7.78-7.78 2.81-2.81c.8-.78.8-2.07 0-2.86zM5.41 20L4 18.59l7.72-7.72 1.47 1.35L5.41 20z\"/>\n",

" </svg>\n",

" </button>\n",

" \n",

" <style>\n",

" .colab-df-container {\n",

" display:flex;\n",

" flex-wrap:wrap;\n",

" gap: 12px;\n",

" }\n",

"\n",

" .colab-df-convert {\n",

" background-color: #E8F0FE;\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: #1967D2;\n",

" height: 32px;\n",

" padding: 0 0 0 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-convert:hover {\n",

" background-color: #E2EBFA;\n",

" box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: #174EA6;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert {\n",

" background-color: #3B4455;\n",

" fill: #D2E3FC;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert:hover {\n",

" background-color: #434B5C;\n",

" box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",

" filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",

" fill: #FFFFFF;\n",

" }\n",

" </style>\n",

"\n",

" <script>\n",

" const buttonEl =\n",

" document.querySelector('#df-88228d97-4393-42c3-a3a5-47a8d1a95afe button.colab-df-convert');\n",

" buttonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

"\n",

" async function convertToInteractive(key) {\n",

" const element = document.querySelector('#df-88228d97-4393-42c3-a3a5-47a8d1a95afe');\n",

" const dataTable =\n",

" await google.colab.kernel.invokeFunction('convertToInteractive',\n",

" [key], {});\n",

" if (!dataTable) return;\n",

"\n",

" const docLinkHtml = 'Like what you see? Visit the ' +\n",

" '<a target=\"\_blank\" href=https://colab.research.google.com/notebooks/data\_table.ipynb>data table notebook</a>'\n",

" + ' to learn more about interactive tables.';\n",

" element.innerHTML = '';\n",

" dataTable['output\_type'] = 'display\_data';\n",

" await google.colab.output.renderOutput(dataTable, element);\n",

" const docLink = document.createElement('div');\n",

" docLink.innerHTML = docLinkHtml;\n",

" element.appendChild(docLink);\n",

" }\n",

" </script>\n",

" </div>\n",

" </div>\n",

" "

]

},

"metadata": {},

"execution\_count": 7

}

]

},

{

"cell\_type": "code",

"source": [

"#Understanding distribution\n",

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

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

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

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 367

},

"id": "t0pIRVsuJC7F",

"outputId": "5cb2a426-86a0-4448-aee6-707ab3ee405b"

},

"execution\_count": 8,

"outputs": [

{

"output\_type": "stream",

"name": "stderr",

"text": [

"/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: 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",

" FutureWarning\n"

]

},

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

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

]

},

"metadata": {},

"execution\_count": 8

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

},

{

"cell\_type": "code",

"source": [

"#training and test data\n",

"X = df.v2\n",

"Y = df.v1\n",

"le = LabelEncoder()\n",

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

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

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

],

"metadata": {

"id": "e9pcKONPJK4x"

},

"execution\_count": 9,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#Processing the data\n",

"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",

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

],

"metadata": {

"id": "WApEaFmMJPwu"

},

"execution\_count": 10,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"Create Model and Add Layers (LSTM, Dense-(Hidden Layers), Output)"

],

"metadata": {

"id": "CcvtUEcpJWsX"

}

},

{

"cell\_type": "code",

"source": [

"def RNN():\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)\n",

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

" return model"

],

"metadata": {

"id": "zEv8GEQyJTuG"

},

"execution\_count": 11,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"Compile the Model"

],

"metadata": {

"id": "5LpKqnexJfNY"

}

},

{

"cell\_type": "code",

"source": [

"model = RNN()\n",

"model.summary()\n",

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

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "loroQOKGJbdm",

"outputId": "64fb4891-91a1-444f-9742-f4e1fc2d4b85"

},

"execution\_count": 12,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"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"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"Fit the Model\n"

],

"metadata": {

"id": "W7ekxmYDJok8"

}

},

{

"cell\_type": "code",

"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)])"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "oaToRO1mJkcJ",

"outputId": "607bec36-50c7-4552-faa2-5894ff0fa844"

},

"execution\_count": 13,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"30/30 [==============================] - 12s 292ms/step - loss: 0.3222 - accuracy: 0.8812 - val\_loss: 0.1987 - val\_accuracy: 0.9198\n",

"Epoch 2/10\n",

"30/30 [==============================] - 8s 270ms/step - loss: 0.0818 - accuracy: 0.9789 - val\_loss: 0.0778 - val\_accuracy: 0.9747\n"

]

},

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<keras.callbacks.History at 0x7fb9f8048f50>"

]

},

"metadata": {},

"execution\_count": 13

}

]

},

{

"cell\_type": "markdown",

"source": [

"Save The Model"

],

"metadata": {

"id": "RJowbuPNJy0y"

}

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "KVK0UJJLJz6p",

"outputId": "45e372e8-7b87-4ab1-879b-e153f496a962"

},

"execution\_count": 14,

"outputs": [

{

"output\_type": "stream",

"name": "stderr",

"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"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"Test The Model"

],

"metadata": {

"id": "lBfYQ\_Y1J9sy"

}

},

{

"cell\_type": "code",

"source": [

"def test():\n",

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

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

"\n",

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

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

"\n",

"test()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "w\_hioK7cJ6kO",

"outputId": "058f1ded-0c79-4b4f-b214-0375b0f9e162"

},

"execution\_count": 15,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"27/27 [==============================] - 1s 43ms/step - loss: 0.0693 - accuracy: 0.9833\n",

"Test set\n",

" Loss: 0.069\n",

" Accuracy: 0.983\n"

]

}

]

}

]

}