{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"\*\*1. Required libararies are imported\*\*"

],

"metadata": {

"id": "UOP6ubuAdGT6"

}

},

{

"cell\_type": "code",

"source": [

"import numpy as np \n",

"import pandas as pd \n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

"import keras\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, pad\_sequences\n",

"from keras.callbacks import EarlyStopping\n",

"%matplotlib inline"

],

"metadata": {

"id": "udon7G0Un5Fe"

},

"execution\_count": 9,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*2. Read dataset and pre processing\*\*"

],

"metadata": {

"id": "ImTr4A\_Vc7gW"

}

},

{

"cell\_type": "code",

"source": [

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

"df.head()"

],

"metadata": {

"colab": {

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

"height": 206

},

"id": "YZlUPesEn\_-K",

"outputId": "901536d3-4ae5-4d5c-b841-35a143e651e9"

},

"execution\_count": 10,

"outputs": [

{

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

"data": {

"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 "

],

"text/html": [

"\n",

" <div id=\"df-1a9347e9-a8a2-4ef0-8276-5f475593bbee\">\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",

" <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>\n",

" <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-1a9347e9-a8a2-4ef0-8276-5f475593bbee')\"\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-1a9347e9-a8a2-4ef0-8276-5f475593bbee 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-1a9347e9-a8a2-4ef0-8276-5f475593bbee');\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": 10

}

]

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "0gTwddcYn\_8u"

},

"execution\_count": 11,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df.shape"

],

"metadata": {

"colab": {

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

},

"id": "87OfQKZaoeLs",

"outputId": "67d9915a-ad0f-4a9e-c8d8-7cc16aff88e5"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"(5572, 2)"

]

},

"metadata": {},

"execution\_count": 12

}

]

},

{

"cell\_type": "code",

"source": [

"#plot the ham and spam messages to understand the distribution\n",

"df['v1'].value\_counts().plot(kind='bar')\n",

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

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

],

"metadata": {

"colab": {

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

"height": 331

},

"id": "Bs-mpiD5o7mx",

"outputId": "c7d04c52-74bf-439f-8299-c196a664cd92"

},

"execution\_count": 12,

"outputs": [

{

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

"data": {

"text/plain": [

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

]

},

"metadata": {},

"execution\_count": 12

},

{

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

"data": {

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

},

{

"cell\_type": "code",

"source": [

"X = df.v2\n",

"Y = df.v1\n",

"#label encoding for Y\n",

"le = LabelEncoder()\n",

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

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

],

"metadata": {

"id": "6U3jJiiWo9pB"

},

"execution\_count": 13,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*3. Train-test split\*\*"

],

"metadata": {

"id": "UzyiPw\_fdXqd"

}

},

{

"cell\_type": "code",

"source": [

"#split into train and test sets\n",

"X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.20)"

],

"metadata": {

"id": "AohH9l9Qo9nO"

},

"execution\_count": 14,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*4. Tokenizer\*\*"

],

"metadata": {

"id": "3YFwJm5RdZiw"

}

},

{

"cell\_type": "code",

"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",

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

],

"metadata": {

"id": "6TMTwpfZo7lQ"

},

"execution\_count": 15,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*5. Add Layers(LSTM, Dense-(Hidden Layers), Output)\*\*"

],

"metadata": {

"id": "upjlOGwwdexI"

}

},

{

"cell\_type": "code",

"source": [

"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)"

],

"metadata": {

"id": "ehHGZkKsdx3I"

},

"execution\_count": 16,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*6. Create Model\*\*"

],

"metadata": {

"id": "WtCoC2JVdjCg"

}

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "Nn0Xk8a\_pRjF"

},

"execution\_count": 17,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*7. Compile the Model\*\*"

],

"metadata": {

"id": "SvqfHZ8UdmkI"

}

},

{

"cell\_type": "code",

"source": [

"model.summary()\n",

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

],

"metadata": {

"colab": {

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

},

"id": "0NlWVacvpRhq",

"outputId": "238c0d65-3139-46aa-fa04-93b44ee07ad4"

},

"execution\_count": 18,

"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": [

"\*\*8.Fit the Model\*\*"

],

"metadata": {

"id": "vqM3krGwdq0B"

}

},

{

"cell\_type": "code",

"source": [

"model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,validation\_split=0.2,callbacks=[EarlyStopping(monitor='val\_loss',min\_delta=0.0001)])"

],

"metadata": {

"colab": {

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

},

"id": "VoJCUH9KpRgA",

"outputId": "9329f288-4ab4-4c3a-f212-d38c0268d498"

},

"execution\_count": 25,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"28/28 [==============================] - 9s 331ms/step - loss: 0.0448 - accuracy: 0.9871 - val\_loss: 0.0516 - val\_accuracy: 0.9854\n",

"Epoch 2/10\n",

"28/28 [==============================] - 7s 244ms/step - loss: 0.0311 - accuracy: 0.9919 - val\_loss: 0.0508 - val\_accuracy: 0.9821\n"

]

},

{

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

"data": {

"text/plain": [

"<keras.callbacks.History at 0x7f39b27bdb90>"

]

},

"metadata": {},

"execution\_count": 25

}

]

},

{

"cell\_type": "markdown",

"source": [

"\*\*9. Save the Model\*\*"

],

"metadata": {

"id": "HSS6PA\_3duow"

}

},

{

"cell\_type": "code",

"source": [

"model.save('spam\_lstm\_model.Dinesh')"

],

"metadata": {

"colab": {

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

},

"id": "VNg\_QwoIpift",

"outputId": "9a3f5150-9a3e-4451-bcb8-e760bf0c35ab"

},

"execution\_count": 20,

"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": [

"\*\*10.Test the Model\*\*"

],

"metadata": {

"id": "YxXcSUpRdykR"

}

},

{

"cell\_type": "code",

"source": [

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

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

],

"metadata": {

"id": "CYkFpkW4pnqF"

},

"execution\_count": 21,

"outputs": []

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"colab": {

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

},

"id": "m5Pjey4RppfN",

"outputId": "7958fed7-363f-4cb1-c34a-13003668fabe"

},

"execution\_count": 22,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"35/35 [==============================] - 2s 51ms/step - loss: 0.0621 - accuracy: 0.9839\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"colab": {

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

},

"id": "rCBZBza5ljEM",

"outputId": "6609b9bd-9f11-4725-82ef-032c199de72c"

},

"execution\_count": 23,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Test set\n",

" Loss: 0.062\n",

" Accuracy: 0.984\n"

]

}

]

}

]

}