

Import the Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving spam.csv to spam.csv

Import required libraries

```
import csv
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
```

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.

Import dataset

```
import io
dataset = pd.read_csv(io.BytesIO(uploaded['spam.csv']), encoding = "ISO-8859-1")
```

dataset

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will l_b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows x 5 columns

```

vocab_size = 5000
embedding_dim = 64
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = '<OOV>'
training_portion = .8

```

[8]

Python

Read the dataset and do pre-processing.

To remove the stop words.

```

articles = []
labels = []

with open("spam.csv", 'r', encoding = "ISO-8859-1") as dataset:
    reader = csv.reader(dataset, delimiter=',')
    next(reader)
    for row in reader:
        labels.append(row[0])
        article = row[1]
        for word in STOPWORDS:
            token = ' ' + word + ' '
            article = article.replace(token, ' ')
            article = article.replace(' ', ' ')
        articles.append(article)
print(len(labels))
print(len(articles))

```

[10]

Python

```

... 5572
    5572

```

Train the model

```

> train_size = int(len(articles) * training_portion)

train_articles = articles[0: train_size]
train_labels = labels[0: train_size]

validation_articles = articles[train_size:]
validation_labels = labels[train_size:]

print(train_size)
print(len(train_articles))
print(len(train_labels))
print(len(validation_articles))
print(len(validation_labels))

```

[11]

Python

```

... 4457
    4457
    4457
    1115
    1115

```

Python

Training data to Sequences

Python

Train neural network for NLP

Python

Python

... 1115
(1115, 200)

```
label_tokenizer = Tokenizer()
label_tokenizer.fit_on_texts(labels)

training_label_seq = np.array(label_tokenizer.texts_to_sequences(train_labels))
validation_label_seq = np.array(label_tokenizer.texts_to_sequences(validation_labels))
print(training_label_seq[0])
print(training_label_seq[1])
print(training_label_seq[2])
print(training_label_seq.shape)

print(validation_label_seq[0])
print(validation_label_seq[1])
print(validation_label_seq[2])
print(validation_label_seq.shape)
```

[17]

Python

```
... [1]
[1]
[2]
(4457, 1)
[1]
[2]
[1]
(1115, 1)
```

```
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])

def decode_article(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])

print(decode_article(train_padded[10]))
print('---')
print(train_articles[10])
```

[18] Python

```
... I'm gonna home soon want talk stuff anymore tonight k i've cried enough today ??????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
?????????????????????????????????????????????????????????????????????????????????????????????????????????????????
```

```
I'm gonna home soon want talk stuff anymore tonight, k? I've cried enough today.
```

To implement LSTM

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embedding_dim)),
    tf.keras.layers.Dense(embedding_dim, activation='relu'),
    tf.keras.layers.Dense(6, activation='softmax')
])
model.summary()
```

[19]

Python

```
... Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 64)	320000
bidirectional (BidirectionalLSTM)	(None, 128)	66048
dense (Dense)	(None, 64)	8256
dense_1 (Dense)	(None, 6)	390

```

Total params: 394,694
Trainable params: 394,694
Non-trainable params: 0

```

```

print(set(labels))

[20] Python

... ('ham', 'spam')

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
num_epochs = 10
history = model.fit(train_padded, training_label_seq, epochs=num_epochs, validation_data=(validation_padded, validation_label_seq), verbose=2)

[21] Python

... Epoch 1/10
140/140 - 33s - loss: 0.3079 - accuracy: 0.9237 - val_loss: 0.0536 - val_accuracy: 0.9874 - 33s/epoch - 239ms/step
Epoch 2/10
140/140 - 30s - loss: 0.0297 - accuracy: 0.9924 - val_loss: 0.0432 - val_accuracy: 0.9874 - 30s/epoch - 213ms/step
Epoch 3/10
140/140 - 31s - loss: 0.0129 - accuracy: 0.9973 - val_loss: 0.0366 - val_accuracy: 0.9901 - 31s/epoch - 220ms/step
Epoch 4/10
140/140 - 31s - loss: 0.0050 - accuracy: 0.9991 - val_loss: 0.0663 - val_accuracy: 0.9821 - 31s/epoch - 224ms/step
Epoch 5/10
140/140 - 28s - loss: 0.0062 - accuracy: 0.9982 - val_loss: 0.0467 - val_accuracy: 0.9892 - 28s/epoch - 203ms/step
Epoch 6/10
140/140 - 29s - loss: 0.0021 - accuracy: 0.9996 - val_loss: 0.0495 - val_accuracy: 0.9874 - 29s/epoch - 206ms/step
Epoch 7/10
140/140 - 30s - loss: 0.0012 - accuracy: 0.9998 - val_loss: 0.0610 - val_accuracy: 0.9892 - 30s/epoch - 216ms/step
Epoch 8/10
140/140 - 32s - loss: 9.7783e-04 - accuracy: 0.9996 - val_loss: 0.0608 - val_accuracy: 0.9848 - 32s/epoch - 229ms/step
Epoch 9/10
140/140 - 31s - loss: 8.1823e-04 - accuracy: 0.9998 - val_loss: 0.0574 - val_accuracy: 0.9848 - 31s/epoch - 219ms/step
Epoch 10/10
140/140 - 30s - loss: 4.3584e-04 - accuracy: 0.9998 - val_loss: 0.0651 - val_accuracy: 0.9848 - 30s/epoch - 215ms/step

```

```

def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_'+string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")

```

Python

