Universal Sentence Encoder on Amazon Reviews

By *Alex Vaillant*

Research Question

"To what extent can we accurately identify our future customer's sentiment in unseen reviews based on the past reviews on our products by using NLP and NN techniques? The end model will be used as part of an early "warning" system to trigger the need for intervention and praise based on the review's sentiment."

Set Up Environment

In [1]:
```
from platform import python_version
print(python_version())
```

```
3.7.10
```

Load Libraries

In [2]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import keras
from keras import layers, Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import load_model
import tensorflow_hub as hub
import tensorflow_text
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
```

Load Universal Sentence Encoder

In [3]:
```
USE = hub.load("https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/
```

Data Extraction

In [4]:
```
URL = 'C:/Users/tedda/Desktop/Data Science Portfolio/Machine Learning/Sentiment Analysi
amazon_data = pd.read_csv(URL, delimiter = '\t', names = ['Review','Review_Type'])
print('Amazon Raw Data Shape:', amazon_data.shape)
```

```
Amazon Raw Data Shape: (1000, 2)
```

Data Preparation

One Hot Encode the Review Labels

In [5]:
```
onehot = OneHotEncoder(sparse=False).fit_transform(amazon_data['Review_Type'].to_numpy(
```

In [6]:
```
onehot
```

Out[6]:
```
array([[1., 0.],
       [0., 1.],
       [0., 1.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

Split data into Training and Testing sets (80%, 20%)

In [7]:
```
x_train, x_test, y_train, y_test = train_test_split(amazon_data['Review'], onehot, test
```

USE the Reviews' Text (X)

In [8]:
```
X_train = []
for review in x_train:
    embedded = USE(review)
    reviews_embedded = tf.reshape(embedded, [-1]).numpy()
    X_train.append(reviews_embedded)

X_train = np.array(X_train)
```

In [9]:
```
X_train.shape
```

Out[9]:
```
(800, 512)
```

In [10]:
```
X_test = []
for review in x_test:
    embedded = USE(review)
    reviews_embedded = tf.reshape(embedded, [-1]).numpy()
    X_test.append(reviews_embedded)

X_test = np.array(X_test)
```

In [11]:
```
X_test.shape
```

Out[11]:
```
(200, 512)
```

Export Cleansed Datasets

In [12]:
```
pd.DataFrame(X_train).to_csv('C:/Users/tedda/Desktop/Data Science Portfolio/Machine Lea
pd.DataFrame(X_test).to_csv('C:/Users/tedda/Desktop/Data Science Portfolio/Machine Lear
pd.DataFrame(y_train).to_csv('C:/Users/tedda/Desktop/Data Science Portfolio/Machine Lea
pd.DataFrame(y_test).to_csv('C:/Users/tedda/Desktop/Data Science Portfolio/Machine Lear
```

Build the Model

In [13]:
```
model = keras.Sequential()

model.add(keras.layers.Dense(256, input_shape=(X_train.shape[1], ), activation = 'relu'

model.add(keras.layers.Dropout(rate=0.5))

model.add(keras.layers.Dense(128, activation = 'relu'))

model.add(keras.layers.Dropout(rate=0.5))

model.add(keras.layers.Dense(2, activation = 'softmax'))

model.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               131328
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 164,482
Trainable params: 164,482
Non-trainable params: 0
_________________________________________________________________
```

Train the Model

In [14]:
```
model.fit(X_train, y_train, epochs = 4, batch_size = 20, validation_split = 0.2, verbos
```

```
Epoch 1/4
32/32 [==============================] - 40s 36ms/step - loss: 0.6711 - accuracy: 0.6349
- val_loss: 0.5480 - val_accuracy: 0.8125
Epoch 2/4
32/32 [==============================] - ETA: 0s - loss: 0.4438 - accuracy: 0.88 - 0s 6m
s/step - loss: 0.4337 - accuracy: 0.8861 - val_loss: 0.3684 - val_accuracy: 0.8625
Epoch 3/4
32/32 [==============================] - 0s 6ms/step - loss: 0.2444 - accuracy: 0.9297 -
val_loss: 0.3626 - val_accuracy: 0.8687
Epoch 4/4
32/32 [==============================] - 0s 6ms/step - loss: 0.1853 - accuracy: 0.9414 -
val_loss: 0.3911 - val_accuracy: 0.8438
```

Out[14]:
```
<keras.callbacks.History at 0x265f3f1b4c8>
```

Save and Load the Model

In [15]:
```
model_url = 'C:/Users/tedda/Desktop/Data Science Portfolio/Machine Learning/Sentiment A
model.save(model_url)
```

```
SA_model = load_model(model_url)
```

Evaluate the Model's Accuracy

In [16]:
```
SA_model.evaluate(X_test, y_test)
```

```
7/7 [==============================] - 1s 4ms/step - loss: 0.1862 - accuracy: 0.9250
```

Out[16]:
```
[0.18624836206436157, 0.925000011920929]
```