# imdb reviews classification with BERT

Muhammad Tamjid Rahman

## Contents



## Loading R packages

```
library(uuml)
```

### Python packages

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import imdb
from tensorflow.keras import preprocessing
from tensorflow.keras.models import Sequential
from keras import layers
from keras import models
import tensorflow_hub as hub
import tensorflow_text as text
from tensorflow.keras import Model
from tensorflow.keras.layers import Flatten, Dense, Dropout, Embedding
from tensorflow.keras.layers import BatchNormalization
import matplotlib.pyplot as plt
```

```
# Number of words to consider as features
max_features = 10000
# First 30 words(among the most frequent 10,000) would be considered of a single review
maxlen = 30
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
x_train = preprocessing.sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = preprocessing.sequence.pad_sequences(x_test, maxlen=maxlen)
```

# 1 Implementing an BERT Model

### Needed Packages

```
# Bert prepossessing layer
bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
```

```
# Coresponding encoding layer
bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

```
def decode_imdb_reviews(text_data):
    result = [0 for x in range(len(text_data))]
    word_index = imdb.get_word_index()
    reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
    for review in range(0,len(text_data)):
        for index in enumerate(text_data[review]):
            decoded_review = ' '.join([reverse_word_index.get(index - 3, '?') for index in text_data[rev
        result[review] = decoded_review
    return result
```

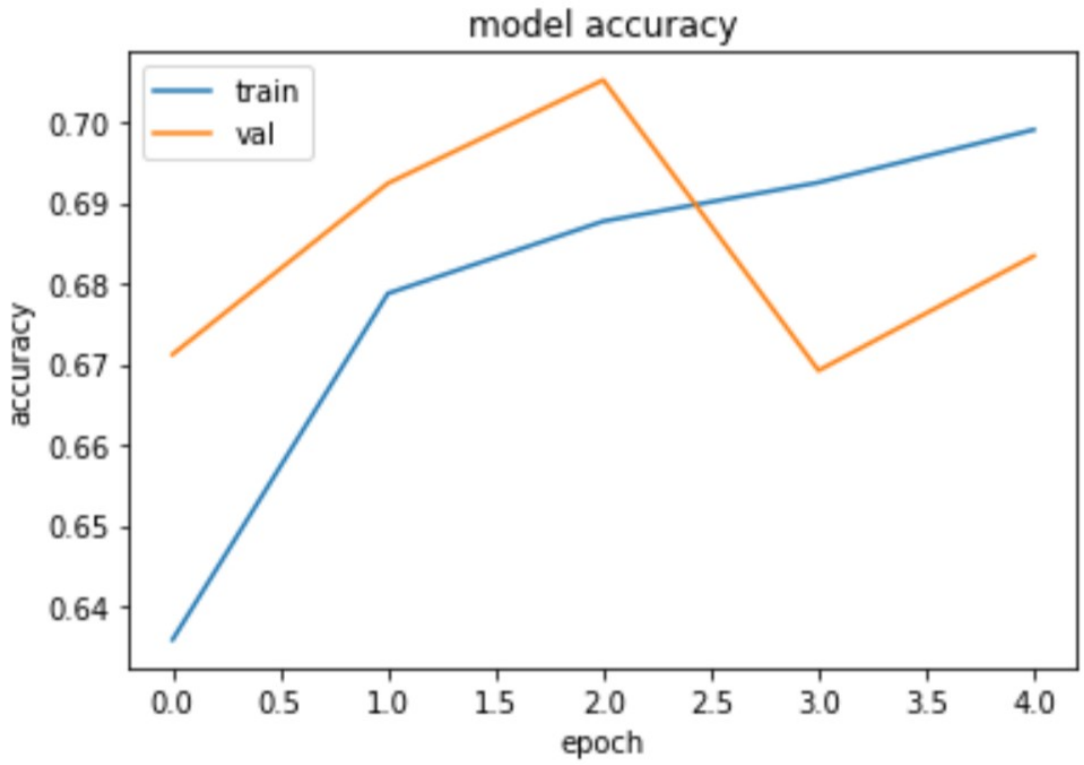
```
x_train1 = decode_imdb_reviews(x_train)
```

```
# Bert layers
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)

# Neural network layers
l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
l = tf.keras.layers.Flatten()(l)
l = tf.keras.layers.Dense(64, activation='sigmoid')(l)
l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)

# Use inputs and outputs to construct a final model
model = tf.keras.Model(inputs=[text_input], outputs = [l])
model.summary()
```

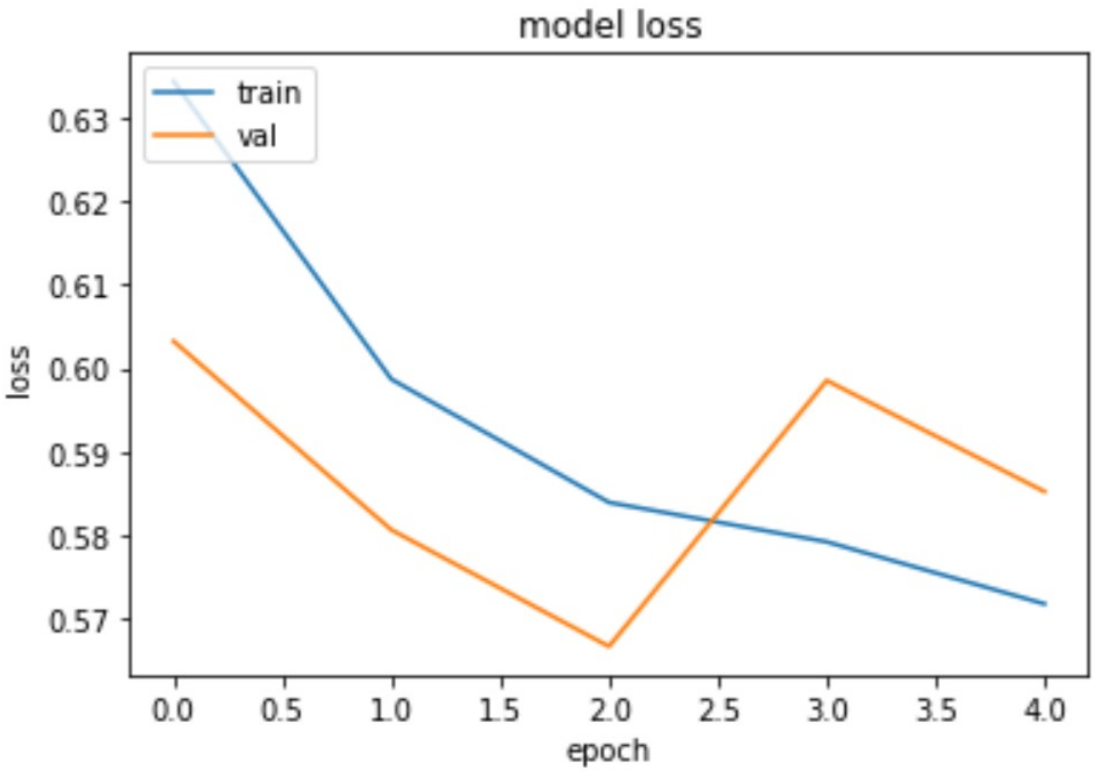
```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 text (InputLayer)              [(None,)]            0           []

 keras_layer (KerasLayer)       {'input_word_ids':   0           ['text[0][0]']
                                (None, 128),
                                 'input_type_ids':
                                (None, 128),
                                 'input_mask': (Non
                                e, 128)}

 keras_layer_1 (KerasLayer)     {'default': (None,   109482241   ['keras_layer[1][0]',
                                768),                             'keras_layer[1][1]',
                                 'encoder_outputs':               'keras_layer[1][2]']
                                 [(None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768)],
                                 'sequence_output':
                                 (None, 128, 768),
                                 'pooled_output': (
                                None, 768)}

 dropout (Dropout)              (None, 768)          0           ['keras_layer_1[1][13]']

 flatten_1 (Flatten)            (None, 768)          0           ['dropout[0][0]']

 dense_1 (Dense)                (None, 64)           49216       ['flatten_1[0][0]']

 output (Dense)                 (None, 1)            65          ['dense_1[0][0]']

==================================================================================================
Total params: 109,531,522
Trainable params: 49,281
Non-trainable params: 109,482,241
```

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])

history = model.fit(x_train, y_train,
                    epochs=10,batch_size=128,
                    validation_split=0.2)
```

```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```




```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```




```
x_test1 = decode_imdb_reviews(x_test)
y_test1 = tf.convert_to_tensor(y_test)
x_test2 = tf.convert_to_tensor(x_test1)

test_loss, test_acc = model.evaluate(x_test, y_test)
```

```
test_loss, test_acc = model.evaluate(x_test2, y_test1)
```

```
test_acc
```

```
0.6913599967956543
```

```
test_loss
```

```
0.5780715942382812
```

In this model we tried to used bert encoding then we used neural networks with 64 units. The samples were prepossessed. We got them back into original form (maxlen= 30). Then we prepossess according to the specific bert encoder. The BERT layer has L=12 hidden layers, a hidden size of H=768, and A=12 attention heads. We got the accuracy 69%.