

```
In [69]: import os
import shutil

import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
from official.nlp import optimization # to create AdamW optimizer

import matplotlib.pyplot as plt

tf.get_logger().setLevel('ERROR')
```

```
In [70]: # url = 'https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz'

# dataset = tf.keras.utils.get_file('aclImdb_v1.tar.gz', url,
#                                     untar=True, cache_dir='.',
#                                     cache_subdir='')

# dataset_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')

# train_dir = os.path.join(dataset_dir, 'train')

# # remove unused folders to make it easier to load the data
# remove_dir = os.path.join(train_dir, 'unsup')
# shutil.rmtree(remove_dir)
```

```
In [71]: AUTOTUNE = tf.data.AUTOTUNE
batch_size = 32
seed = 42

raw_train_ds = tf.keras.utils.text_dataset_from_directory(
    'aclImdb/train',
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=seed)

class_names = raw_train_ds.class_names
train_ds = raw_train_ds.cache().prefetch(buffer_size=AUTOTUNE)

val_ds = tf.keras.utils.text_dataset_from_directory(
    'aclImdb/train',
    batch_size=batch_size,
    validation_split=0.2,
    subset='validation',
    seed=seed)

val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

test_ds = tf.keras.utils.text_dataset_from_directory(
    'aclImdb/test',
    batch_size=batch_size)

test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Found 25000 files belonging to 2 classes.
 Using 20000 files for training.
 Found 25000 files belonging to 2 classes.
 Using 5000 files for validation.
 Found 25000 files belonging to 2 classes.

Let's take a look at a few reviews.

In [72]: *#@title Choose a BERT model to fine-tune*

```
bert_model_name = 'small_bert/bert_en_uncased_L-4_H-512_A-8' #@param ["bert_e

map_name_to_handle = {
  'bert_en_uncased_L-12_H-768_A-12':
    'https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/3',
  'bert_en_cased_L-12_H-768_A-12':
    'https://tfhub.dev/tensorflow/bert_en_cased_L-12_H-768_A-12/3',
  'bert_multi_cased_L-12_H-768_A-12':
    'https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/3',
  'small_bert/bert_en_uncased_L-2_H-128_A-2':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-2_H-128_A-2',
  'small_bert/bert_en_uncased_L-2_H-256_A-4':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-2_H-256_A-4',
  'small_bert/bert_en_uncased_L-2_H-512_A-8':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-2_H-512_A-8',
  'small_bert/bert_en_uncased_L-2_H-768_A-12':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-2_H-768_A-1',
  'small_bert/bert_en_uncased_L-4_H-128_A-2':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-128_A-2',
  'small_bert/bert_en_uncased_L-4_H-256_A-4':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-256_A-4',
  'small_bert/bert_en_uncased_L-4_H-512_A-8':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8',
  'small_bert/bert_en_uncased_L-4_H-768_A-12':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-768_A-1',
  'small_bert/bert_en_uncased_L-6_H-128_A-2':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-6_H-128_A-2',
  'small_bert/bert_en_uncased_L-6_H-256_A-4':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-6_H-256_A-4',
  'small_bert/bert_en_uncased_L-6_H-512_A-8':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-6_H-512_A-8',
  'small_bert/bert_en_uncased_L-6_H-768_A-12':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-6_H-768_A-1',
  'small_bert/bert_en_uncased_L-8_H-128_A-2':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-8_H-128_A-2',
  'small_bert/bert_en_uncased_L-8_H-256_A-4':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-8_H-256_A-4',
  'small_bert/bert_en_uncased_L-8_H-512_A-8':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-8_H-512_A-8',
  'small_bert/bert_en_uncased_L-8_H-768_A-12':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-8_H-768_A-1',
  'small_bert/bert_en_uncased_L-10_H-128_A-2':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-10_H-128_A-2',
  'small_bert/bert_en_uncased_L-10_H-256_A-4':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-10_H-256_A-4',
  'small_bert/bert_en_uncased_L-10_H-512_A-8':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-10_H-512_A-8',
  'small_bert/bert_en_uncased_L-10_H-768_A-12':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-10_H-768_A-
```

```

'small_bert/bert_en_uncased_L-12_H-128_A-2':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-128_A-
'small_bert/bert_en_uncased_L-12_H-256_A-4':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-256_A-
'small_bert/bert_en_uncased_L-12_H-512_A-8':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-512_A-
'small_bert/bert_en_uncased_L-12_H-768_A-12':
    'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-
'albert_en_base':
    'https://tfhub.dev/tensorflow/albert_en_base/2',
'electra_small':
    'https://tfhub.dev/google/electra_small/2',
'electra_base':
    'https://tfhub.dev/google/electra_base/2',
'experts_pubmed':
    'https://tfhub.dev/google/experts/bert/pubmed/2',
'experts_wiki_books':
    'https://tfhub.dev/google/experts/bert/wiki_books/2',
'talking-heads_base':
    'https://tfhub.dev/tensorflow/talkheads_ggelu_bert_en_base/1',
}

map_model_to_preprocess = {
    'bert_en_uncased_L-12_H-768_A-12':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'bert_en_cased_L-12_H-768_A-12':
        'https://tfhub.dev/tensorflow/bert_en_cased_preprocess/3',
    'small_bert/bert_en_uncased_L-2_H-128_A-2':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-2_H-256_A-4':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-2_H-512_A-8':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-2_H-768_A-12':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-4_H-128_A-2':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-4_H-256_A-4':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-4_H-512_A-8':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-4_H-768_A-12':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-6_H-128_A-2':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-6_H-256_A-4':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-6_H-512_A-8':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-6_H-768_A-12':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-8_H-128_A-2':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-8_H-256_A-4':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-8_H-512_A-8':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-8_H-768_A-12':
        'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
    'small_bert/bert_en_uncased_L-10_H-128_A-2':

```

```

'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-10_H-256_A-4':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-10_H-512_A-8':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-10_H-768_A-12':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-12_H-128_A-2':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-12_H-256_A-4':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-12_H-512_A-8':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'small_bert/bert_en_uncased_L-12_H-768_A-12':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'bert_multi_cased_L-12_H-768_A-12':
'https://tfhub.dev/tensorflow/bert_multi_cased_preprocess/3',
'albert_en_base':
'https://tfhub.dev/tensorflow/albert_en_preprocess/3',
'electra_small':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'electra_base':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'experts_pubmed':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'experts_wiki_books':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
'talking-heads_base':
'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3',
}

```

```

tfhub_handle_encoder = map_name_to_handle[bert_model_name]
tfhub_handle_preprocess = map_model_to_preprocess[bert_model_name]

```

```

print(f'BERT model selected          : {tfhub_handle_encoder}')
print(f'Preprocess model auto-selected: {tfhub_handle_preprocess}')

```

```

BERT model selected          : https://tfhub.dev/tensorflow/small_bert/bert_e
n_uncased_L-4_H-512_A-8/1
Preprocess model auto-selected: https://tfhub.dev/tensorflow/bert_en_uncased_p
reprocess/3

```

```
In [73]: bert_preprocess_model = hub.KerasLayer(tfhub_handle_preprocess)
```

```

In [74]: text_test = ['this is such an amazing movie!']
text_preprocessed = bert_preprocess_model(text_test)

print(f'Keys          : {list(text_preprocessed.keys())}')
print(f'Shape         : {text_preprocessed["input_word_ids"].shape}')
print(f'Word Ids        : {text_preprocessed["input_word_ids"][0, :12]}')
print(f'Input Mask       : {text_preprocessed["input_mask"][0, :12]}')
print(f'Type Ids         : {text_preprocessed["input_type_ids"][0, :12]}')

```

```

Keys          : ['input_type_ids', 'input_word_ids', 'input_mask']
Shape         : (1, 128)
Word Ids      : [ 101 2023 2003 2107 2019 6429 3185  999  102    0    0    0]
Input Mask    : [1 1 1 1 1 1 1 1 1 0 0 0]
Type Ids      : [0 0 0 0 0 0 0 0 0 0 0 0]

```

```
In [75]: bert_model = hub.KerasLayer(tfhub_handle_encoder)
```

```
In [76]: bert_results = bert_model(text_preprocessed)

print(f'Loaded BERT: {tfhub_handle_encoder}')
print(f'Pooled Outputs Shape:{bert_results["pooled_output"].shape}')
print(f'Pooled Outputs Values:{bert_results["pooled_output"][0, :12]}')
print(f'Sequence Outputs Shape:{bert_results["sequence_output"].shape}')
print(f'Sequence Outputs Values:{bert_results["sequence_output"][0, :12]}')

Loaded BERT: https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8/1
Pooled Outputs Shape:(1, 512)
Pooled Outputs Values:[ 0.76262873  0.99280983 -0.18611865  0.3667382  0.1523
3754  0.6550446
 0.9681154 -0.948627  0.00216129 -0.9877731  0.06842719 -0.9763059 ]
Sequence Outputs Shape:(1, 128, 512)
Sequence Outputs Values:[[-0.2894631  0.3432125  0.33231503 ... 0.21300879
0.71020824
-0.05771083]
[-0.28741956 0.31981033 -0.23018478 ... 0.58455044 -0.21329702
0.72692096]
[-0.66156983 0.68876874 -0.87432986 ... 0.10877332 -0.26173213
0.4785546 ]
...
[-0.22561109 -0.2892557 -0.07064363 ... 0.47566098 0.8327722
0.40025353]
[-0.29824188 -0.27473086 -0.05450502 ... 0.48849785 1.0955354
0.18163365]
[-0.4437817  0.00930776 0.07223777 ... 0.17290121 1.1833248
0.07898013]]
```

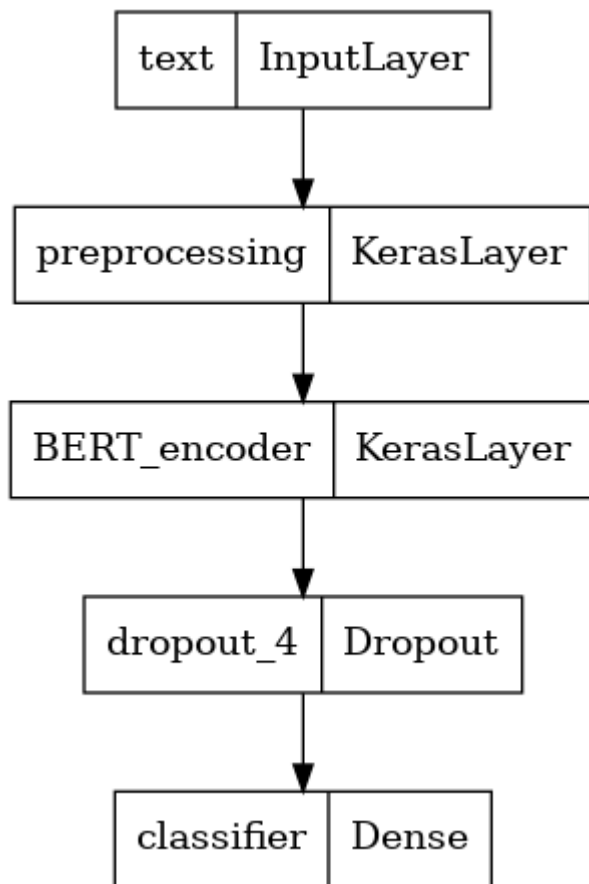
```
In [77]: def build_classifier_model():
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessing_layer = hub.KerasLayer(tfhub_handle_preprocess, name='preproce
encoder_inputs = preprocessing_layer(text_input)
encoder = hub.KerasLayer(tfhub_handle_encoder, trainable=True, name='BERT_en
outputs = encoder(encoder_inputs)
net = outputs['pooled_output']
net = tf.keras.layers.Dropout(0.3)(net)
net = tf.keras.layers.Dense(1, activation=None, name='classifier')(net)
return tf.keras.Model(text_input, net)
```

```
In [78]: classifier_model = build_classifier_model()
bert_raw_result = classifier_model(tf.constant(text_test))
print(tf.sigmoid(bert_raw_result))

tf.Tensor([[0.46414578]], shape=(1, 1), dtype=float32)
```

```
In [79]: tf.keras.utils.plot_model(classifier_model)
```

Out[79]:



```
In [80]: loss = tf.keras.losses.BinaryCrossentropy(from_logits=True)
metrics = tf.metrics.BinaryAccuracy()
```

```
In [81]: epochs = 5
steps_per_epoch = tf.data.experimental.cardinality(train_ds).numpy()
num_train_steps = steps_per_epoch * epochs
num_warmup_steps = int(0.1*num_train_steps)

init_lr = 3e-5
optimizer = optimization.create_optimizer(init_lr=init_lr,
                                         num_train_steps=num_train_steps,
                                         num_warmup_steps=num_warmup_steps,
                                         optimizer_type='adamw')
```

```
In [82]: classifier_model.compile(optimizer=optimizer,
                                loss=loss,
                                metrics=metrics)
```

```
In [83]: print(f'Training model with {tfhub_handle_encoder}')
history = classifier_model.fit(x=train_ds,
                              validation_data=val_ds,
                              epochs=epochs)
```

Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8/1

Epoch 1/5

625/625 [=====] - 149s 233ms/step - loss: 0.4956 - binary_accuracy: 0.7454 - val_loss: 0.4056 - val_binary_accuracy: 0.7992

Epoch 2/5

625/625 [=====] - 148s 237ms/step - loss: 0.3423 - binary_accuracy: 0.8478 - val_loss: 0.3710 - val_binary_accuracy: 0.8354

Epoch 3/5

625/625 [=====] - 149s 238ms/step - loss: 0.2612 - binary_accuracy: 0.8885 - val_loss: 0.3942 - val_binary_accuracy: 0.8422

Epoch 4/5

625/625 [=====] - 148s 237ms/step - loss: 0.1999 - binary_accuracy: 0.9223 - val_loss: 0.4299 - val_binary_accuracy: 0.8496

Epoch 5/5

625/625 [=====] - 149s 239ms/step - loss: 0.1615 - binary_accuracy: 0.9395 - val_loss: 0.4752 - val_binary_accuracy: 0.8480

In [84]: `loss, accuracy = classifier_model.evaluate(test_ds)`

```
print(f'Loss: {loss}')
print(f'Accuracy: {accuracy}')
```

782/782 [=====] - 72s 93ms/step - loss: 0.4578 - binary_accuracy: 0.8557

Loss: 0.45780137181282043

Accuracy: 0.8557199835777283

In [85]: `history_dict = history.history`
`print(history_dict.keys())`

```
acc = history_dict['binary_accuracy']
val_acc = history_dict['val_binary_accuracy']
loss = history_dict['loss']
val_loss = history_dict['val_loss']

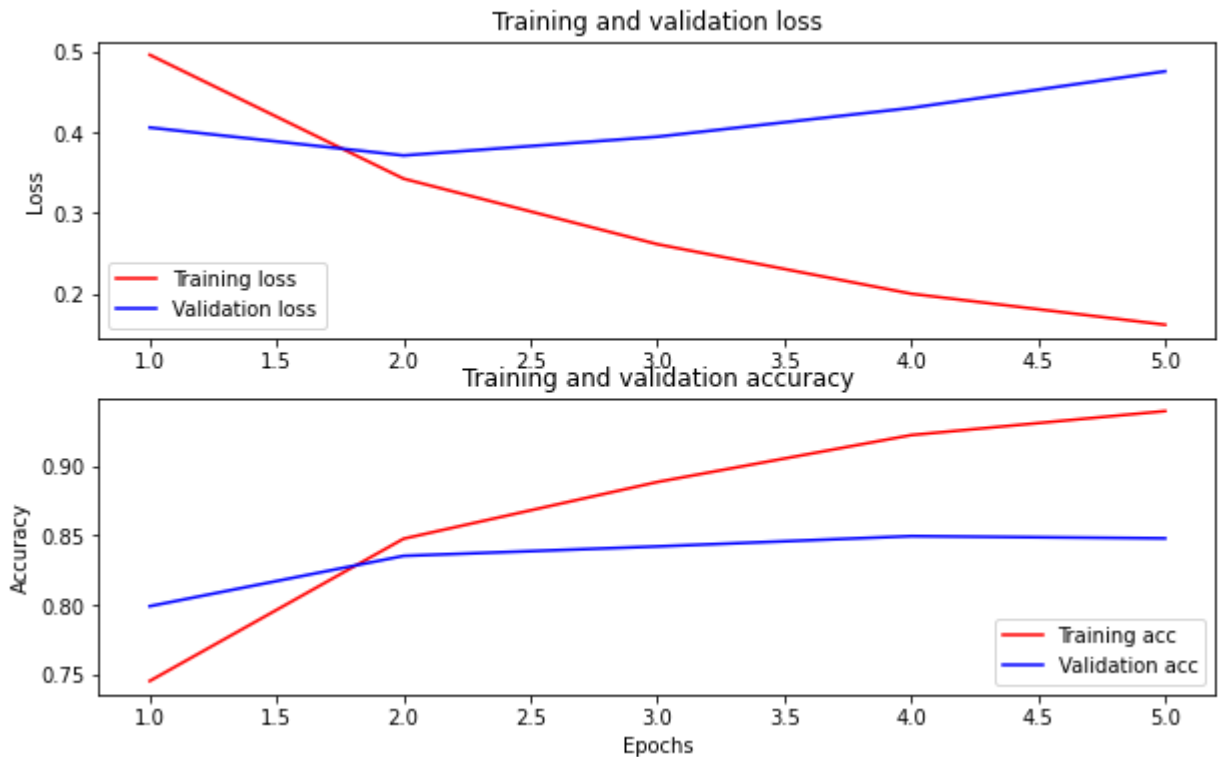
epochs = range(1, len(acc) + 1)
fig = plt.figure(figsize=(10, 6))
fig.tight_layout()

plt.subplot(2, 1, 1)
# r is for "solid red line"
plt.plot(epochs, loss, 'r', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
# plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
```

`dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accuracy'])`

Out[85]: `<matplotlib.legend.Legend at 0x7fb119a6a0d0>`



```
In [86]: dataset_name = 'imdb'
saved_model_path = './{}_bert'.format(dataset_name.replace('/', '_'))

classifier_model.save(saved_model_path, include_optimizer=False)
```

WARNING:absl:Found untraced functions such as restored_function_body, restored_function_body, restored_function_body, restored_function_body, restored_function_body while saving (showing 5 of 124). These functions will not be directly callable after loading.

```
In [87]: reloaded_model = tf.saved_model.load(saved_model_path)
```

```
In [88]: def print_my_examples(inputs, results):
    result_for_printing = \
        [f'input: {inputs[i]:<30} : score: {results[i][0]:.6f}'
          for i in range(len(inputs))]
    print(*result_for_printing, sep='\n')
    print()

    examples = [
        'this is such an amazing movie!', # this is the same sentence tried earlier
        'The movie was great!',
        'The movie was meh.',
        'The movie was okish.',
        'The movie was terrible...'
    ]

    reloaded_results = tf.sigmoid(reloaded_model(tf.constant(examples)))
    original_results = tf.sigmoid(classifier_model(tf.constant(examples)))

    print('Results from the saved model:')
    print_my_examples(examples, reloaded_results)
```



```
print('Results from the model in memory:')  
print_my_examples(examples, original_results)
```

Results from the saved model:

```
input: this is such an amazing movie! : score: 0.999568  
input: The movie was great!           : score: 0.989794  
input: The movie was meh.              : score: 0.992612  
input: The movie was okish.            : score: 0.156652  
input: The movie was terrible...       : score: 0.002084
```

Results from the model in memory:

```
input: this is such an amazing movie! : score: 0.999568  
input: The movie was great!           : score: 0.989794  
input: The movie was meh.              : score: 0.992612  
input: The movie was okish.            : score: 0.156652  
input: The movie was terrible...       : score: 0.002084
```

```
In [89]: serving_results = reloaded_model \  
        .signatures['serving_default'](tf.constant(examples))
```

```
serving_results = tf.sigmoid(serving_results['classifier'])
```

```
print_my_examples(examples, serving_results)
```

```
input: this is such an amazing movie! : score: 0.999568  
input: The movie was great!           : score: 0.989794  
input: The movie was meh.              : score: 0.992612  
input: The movie was okish.            : score: 0.156652  
input: The movie was terrible...       : score: 0.002084
```

```
In [ ]:
```