By: Irene Too
Upload date: 20 March 2022

This notebook is slightly modified from Coursera Guided Project:
Fine-tune a BERT model for text classification using TensorFlow and TF-Hub.

https://www.coursera.org/projects/fine-tune-bert-tensorflow/ (https://www.coursera.org/projects/fine-tune-bert-tensorflow/)

Fine-Tune BERT for Text Classification with TensorFlow

The pretrained BERT model used in this project is available (https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/2) on TensorFlow Hub (https://tfhub.dev/).

Whats in this notebook:

- Build TensorFlow Input Pipelines for Text Data with the `tf.data` (https://www.tensorflow.org/api_docs/python/tf/data) API
- Tokenize and Preprocess Text for BERT
- Fine-tune BERT for text classification with TensorFlow 2 and TF Hub (https://tfhub.dev)

Prerequisites

In order to be successful with this project, it is assumed you are:

- Familiar with deep learning for Natural Language Processing (NLP)
- Familiar with TensorFlow, and its Keras API

Task 2: Setup your TensorFlow and Colab Runtime.

In [ ]:
```
!nvidia-smi
```

```
Sat Mar 19 15:45:40 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   38C    P8    28W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Install TensorFlow and TensorFlow Model Garden

In [ ]:
```
import tensorflow as tf
print(tf.version.VERSION)
```

```
2.8.0
```

In [ ]:
```
!git clone --depth 1 -b v2.3.0 https://github.com/tensorflow/models.git
```

In [ ]:
```
# install requirements to use tensorflow/models repository
!pip install -Uqr models/official/requirements.txt
# you may have to restart the runtime afterwards
```

Task 3: Download and Import the Quora Insincere Questions Dataset

In [ ]:
```
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
import sys
sys.path.append('models')
from official.nlp.data import classifier_data_lib
from official.nlp.bert import tokenization
from official.nlp import optimization
```

In [ ]:
```
# run without gpu
print("TF Version: ", tf.__version__)
print("Eager mode: ", tf.executing_eagerly())
print("Hub version: ", hub.__version__)
print("GPU is", "available" if tf.config.experimental.list_physical_devices("GPU") else "NOT AVAILABLE")
```

```
TF Version:  2.8.0
Eager mode:  True
Hub version:  0.12.0
GPU is available
```

A downloadable copy of the Quora Insincere Questions Classification data (https://www.kaggle.com/c/quora-insincere-questions-classification/data) can be found https://archive.org/download/fine-tune-bert-tensorflow-train.csv/train.csv.zip (https://archive.org/download/fine-tune-bert-tensorflow-train.csv/train.csv.zip). Decompress and read the data into a pandas DataFrame.

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('https://archive.org/download/fine-tune-bert-tensorflow-train.csv/train.csv.zip',
                 compression='zip', low_memory=False)
```

In [ ]:
```
df
```

Out[ ]:

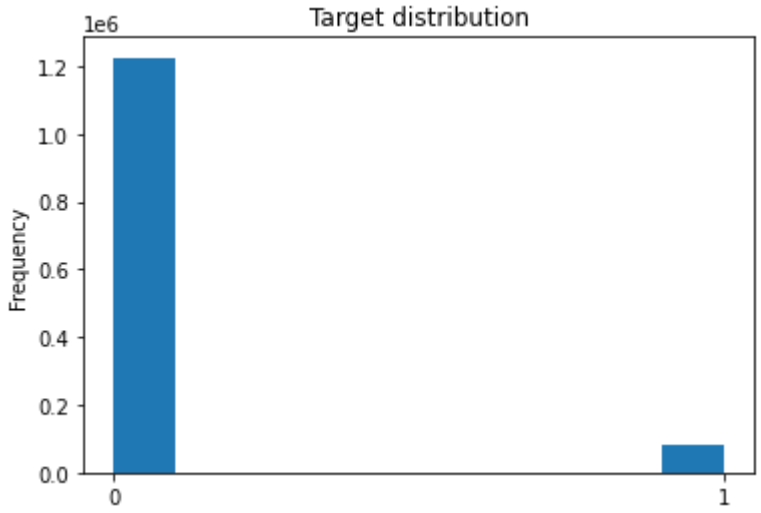
	qid	question_text	target
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0
...	...	...	...
1306117	ffffcc4e2331aaf1e41e	What other technical skills do you need as a c...	0
1306118	ffffd431801e5a2f4861	Does MS in ECE have good job prospects in USA ...	0
1306119	ffffd48fb36b63db010c	Is foam insulation toxic?	0
1306120	fffec519fa37cf60c78	How can one start a research project based on ...	0
1306121	ffffed09fedb5088744a	Who wins in a battle between a Wolverine and a...	0

1306122 rows × 3 columns

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
qid              0
question_text    0
target           0
dtype: int64
```

In [ ]:
```
df.target.plot(kind='hist', title='Target distribution');
```



Task 4: Create tf.data.Datasets for Training and Evaluation

In [ ]:
```
train_df, remaining = train_test_split(df, random_state=42,
                                       train_size=0.009,
                                       stratify=df.target.values)
valid_df, _ = train_test_split(remaining, random_state=42,
                               train_size=0.0009,
                               stratify=remaining.target.values)
```

In [ ]:
```
with tf.device('/cpu:0'):
  train_data = tf.data.Dataset.from_tensor_slices((train_df['question_text'].values,
                                                   train_df['target'].values))

  valid_data = tf.data.Dataset.from_tensor_slices((valid_df['question_text'].values,
                                                   valid_df['target'].values))
  for text, label in train_data.take(1):
    print(text)
    print(label)
```

```
tf.Tensor(b"When will Trump voters realize they've been duped, and who will realize it first?", shape=(), dty
pe=string)

tf.Tensor(1, shape=(), dtype=int64)
```

Task 5: Download a Pre-trained BERT Model from TensorFlow Hub

In [ ]:
```
"""
Each line of the dataset is composed of the review text and its label
- Data preprocessing consists of transforming text to BERT input features:
input_word_ids, input_mask, segment_ids
- In the process, tokenizing the text is done with the provided BERT model tokenizer
"""

# label categories
label_list = [0, 1]
# maximum length of (token) input sequences
max_seq_length = 128
train_batch_size = 32
# Get BERT layer and tokenizer:
bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/2",
                            trainable=True)

# More details here: https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/2
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
tokenizer = tokenization.FullTokenizer(vocab_file,do_lower_case)
```

In [ ]:
```
tokenizer.wordpiece_tokenizer.tokenize('hi, how are you doing?')
```

Out[ ]:
```
['hi', '##,', 'how', 'are', 'you', 'doing', '##?']
```

In [ ]:
```
tokenizer.convert_tokens_to_ids(tokenizer.wordpiece_tokenizer.tokenize('hi, how are you doing?'))
```

Out[ ]:
```
[7632, 29623, 2129, 2024, 2017, 2725, 29632]
```

Task 6: Tokenize and Preprocess Text for BERT

We'll need to transform our data into a format BERT understands. This involves two steps. First, we create InputExamples using `classifier_data_lib` 's constructor `InputExample` provided in the BERT library.

In [ ]:
```
# This provides a function to convert row to input features and label

def to_feature(text, label, label_list=label_list, max_seq_length=max_seq_length, tokenizer=tokenizer):

  example = classifier_data_lib.InputExample(guid=None,
                                             text_a = text.numpy(),
                                             text_b = None,
                                             label = label.numpy())
  feature = classifier_data_lib.convert_single_example(0,example,
                                label_list,max_seq_length,tokenizer)

  return (feature.input_ids, feature.input_mask, feature.segment_ids, feature.label_id)
```

You want to use `Dataset.map` (https://www.tensorflow.org/api_docs/python/tf/data/Dataset#map) to apply this function to each element of the dataset. `Dataset.map` (https://www.tensorflow.org/api_docs/python/tf/data/Dataset#map) runs in graph mode.

- Graph tensors do not have a value.
- In graph mode you can only use TensorFlow Ops and functions.

So you can't `.map` this function directly: You need to wrap it in a `tf.py_function` (https://www.tensorflow.org/api_docs/python/tf/py_function). The `tf.py_function` (https://www.tensorflow.org/api_docs/python/tf/py_function) will pass regular tensors (with a value and a `.numpy()` method to access it), to the wrapped python function.

Task 7: Wrap a Python Function into a TensorFlow op for Eager Execution

In [ ]:
```
def to_feature_map(text, label):
  input_ids, input_mask, segment_ids, label_id = tf.py_function(to_feature, inp=[text, label],
                            Tout = [tf.int32, tf.int32, tf.int32, tf.int32])

  input_ids.set_shape([max_seq_length])
  input_mask.set_shape([max_seq_length])
  segment_ids.set_shape([max_seq_length])
  label_id.set_shape([])

  x = {'input_word_ids': input_ids,
       'input_mask'    : input_mask,
       'input_type_ids' : segment_ids}

  return (x, label_id)
```

Task 8: Create a TensorFlow Input Pipeline with `tf.data`

In [ ]:
```
with tf.device('/cpu:0'):
  # train
  train_data = (train_data.map(to_feature_map,
                               num_parallel_calls=tf.data.experimental.AUTOTUNE)
                          #.cache()
                          .shuffle(1000)
                          .batch(32, drop_remainder=True)
                          .prefetch(tf.data.experimental.AUTOTUNE))

  # valid
  valid_data = (valid_data.map(to_feature_map,
                               num_parallel_calls=tf.data.experimental.AUTOTUNE)
                          .batch(32, drop_remainder=True)
                          .prefetch(tf.data.experimental.AUTOTUNE))
```

The resulting `tf.data.Datasets` return `(features, labels)` pairs, as expected by `keras.Model.fit` (https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit):

In [ ]:
```
# data spec
train_data.element_spec
```

Out[ ]:
```
({'input_mask': TensorSpec(shape=(32, 128), dtype=tf.int32, name=None),
  'input_type_ids': TensorSpec(shape=(32, 128), dtype=tf.int32, name=None),
  'input_word_ids': TensorSpec(shape=(32, 128), dtype=tf.int32, name=None)},
 TensorSpec(shape=(32,), dtype=tf.int32, name=None))
```

Task 9: Add a Classification Head to the BERT Layer

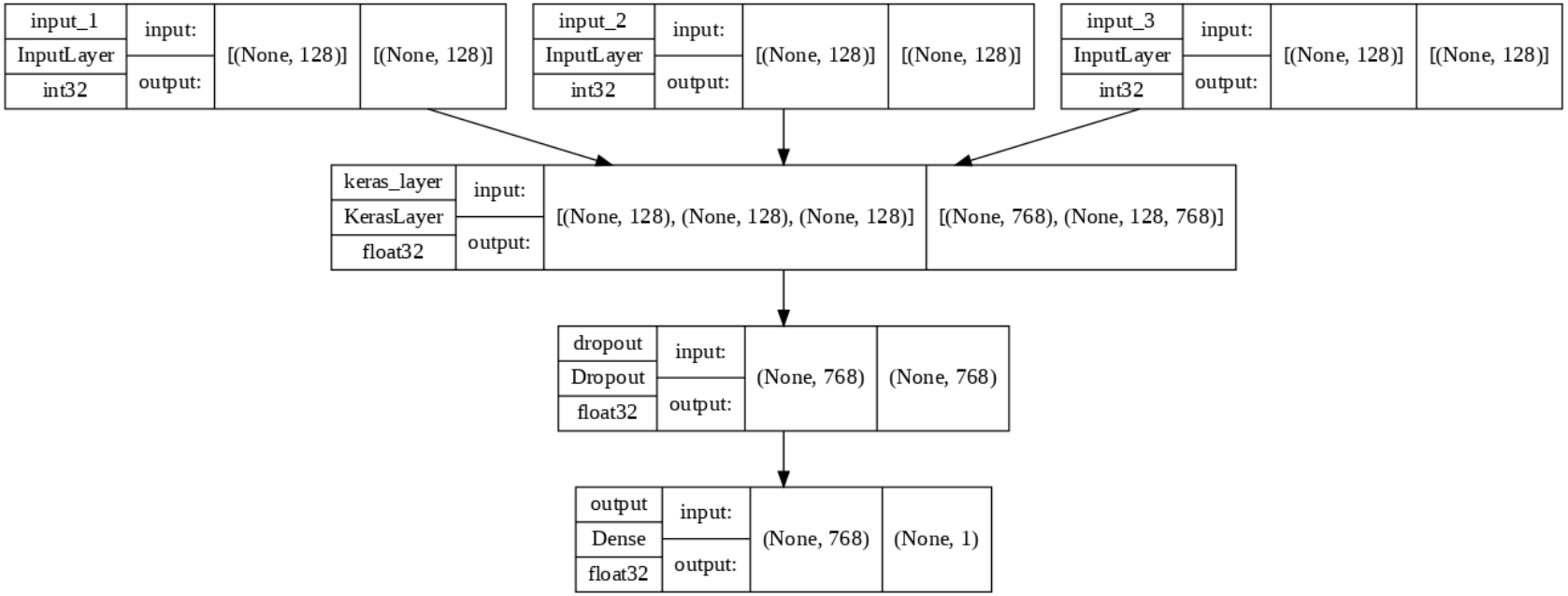
In [ ]:
```
# Building the model
def create_model():
  input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
                                         name="input_word_ids")
  input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
                                     name="input_mask")
  input_type_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
                                     name="input_type_ids")

  pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, input_type_ids])

  drop = tf.keras.layers.Dropout(0.4)(pooled_output)
  output = tf.keras.layers.Dense(1, activation="sigmoid", name="output")(drop)

  model = tf.keras.Model(inputs={'input_word_ids': input_word_ids,
                                 'input_mask': input_mask,
                                 'input_type_ids': input_type_ids},
                         outputs=output)
  return model
```

Task 10: Fine-Tune BERT for Text Classification

In [ ]:
```
model = create_model()
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 2e-4),
              loss = tf.keras.losses.BinaryCrossentropy(),
              metrics = [tf.keras.metrics.BinaryAccuracy()])
# model.summary()
```

In [ ]:
```
tf.keras.utils.plot_model(model=model, show_shapes=True, dpi=90)
```

Out[ ]:



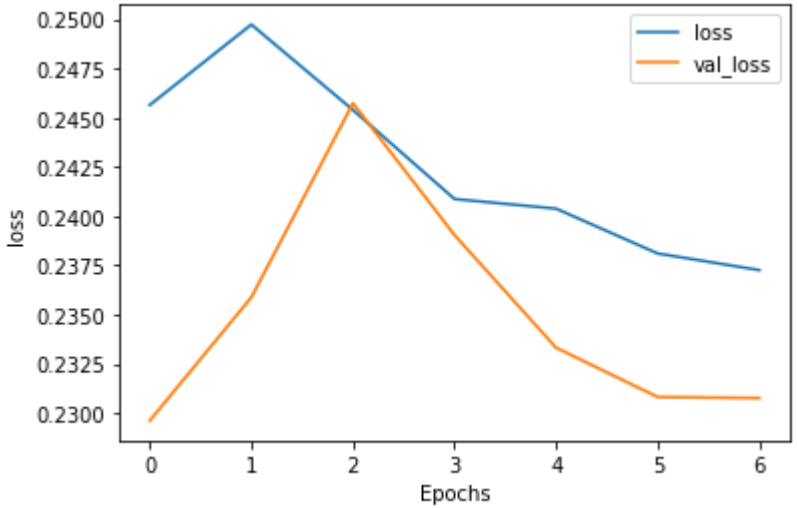
In [ ]:
```
epochs=7
history = model.fit(train_data,
                    validation_data = valid_data,
                    epochs = epochs,
                    verbose= 1)
```

```
Epoch 1/7
367/367 [==============================] - 333s 865ms/step - loss: 0.2456 - binary_accuracy: 0.9355 - val_los
s: 0.2296 - val_binary_accuracy: 0.9392
Epoch 2/7
367/367 [==============================] - 319s 866ms/step - loss: 0.2497 - binary_accuracy: 0.9381 - val_los
s: 0.2359 - val_binary_accuracy: 0.9384
Epoch 3/7
367/367 [==============================] - 318s 865ms/step - loss: 0.2454 - binary_accuracy: 0.9381 - val_los
s: 0.2457 - val_binary_accuracy: 0.9392
Epoch 4/7
367/367 [==============================] - 318s 865ms/step - loss: 0.2409 - binary_accuracy: 0.9382 - val_los
s: 0.2390 - val_binary_accuracy: 0.9392
Epoch 5/7
367/367 [==============================] - 318s 865ms/step - loss: 0.2404 - binary_accuracy: 0.9382 - val_los
s: 0.2333 - val_binary_accuracy: 0.9392
Epoch 6/7
367/367 [==============================] - 319s 866ms/step - loss: 0.2381 - binary_accuracy: 0.9381 - val_los
s: 0.2308 - val_binary_accuracy: 0.9392
Epoch 7/7
367/367 [==============================] - 319s 866ms/step - loss: 0.2373 - binary_accuracy: 0.9381 - val_los
s: 0.2308 - val_binary_accuracy: 0.9392
```

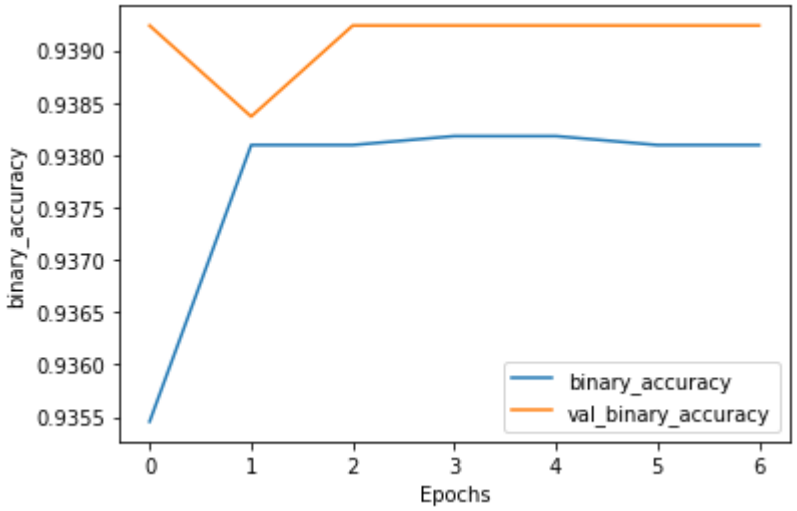
Task 11: Evaluate the BERT Text Classification Model

In [ ]:
```
import matplotlib.pyplot as plt

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

In [ ]:
```
plot_graphs(history, 'loss')
```



In [ ]:
```
plot_graphs(history, 'binary_accuracy')
```



In [ ]:
```
sample_example = ["sentence 1","sentence 2 here"]
test_data = tf.data.Dataset.from_tensor_slices((sample_example, [0]*len(sample_example)))
test_data = (test_data.map(to_feature_map).batch(1))
preds = model.predict(test_data)
res = ['Insincere' if pred >= 0.5 else 'Sincere' for pred in preds]
```

In [ ]:
```
res
```

Out[ ]:
```
['Sincere', 'Sincere']
```