Setup

In [3]:
```
# A dependency of the preprocessing for BERT inputs
!pip install -q tensorflow-text
```

```
     |████████████████████████████████| 3.4MB 3.0MB/s
```

In [4]:
```
# Using AdamW optimizer
!pip install -q tf-models-official==2.4
```

```
     |████████████████████████████████| 1.1MB 2.9MB/s
     |████████████████████████████████| 38.2MB 83kB/s
     |████████████████████████████████| 358kB 36.3MB/s
     |████████████████████████████████| 51kB 6.5MB/s
     |████████████████████████████████| 102kB 9.7MB/s
     |████████████████████████████████| 686kB 18.1MB/s
     |████████████████████████████████| 1.2MB 34.6MB/s
     |████████████████████████████████| 645kB 33.5MB/s
     |████████████████████████████████| 174kB 39.2MB/s
  Building wheel for seqeval (setup.py) ... done
  Building wheel for py-cpuinfo (setup.py) ... done
```

In [5]:
```
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 [6]:
```
url = 'https://github.com/ahlraf/point/blob/main/v1_emails.tar.gz?raw=true'
dataset = tf.keras.utils.get_file('v1_emails.tar.gz', url,
                                  untar=True, cache_dir='.',
                                  cache_subdir='')
```

```
Downloading data from https://github.com/ahlraf/point/blob/main/v1_emails.tar.gz?raw
=true
475136/467631 [==============================] - 0s 0us/step
```

In [7]:
```
dataset_dir = os.path.join(os.path.dirname(dataset), 'v1_emails')
train_dir = os.path.join(dataset_dir, 'train')
```

In [8]:
```
AUTOTUNE = tf.data.AUTOTUNE
batch_size = 32
seed = 42

raw_train_ds = tf.keras.preprocessing.text_dataset_from_directory(
    'v1_emails/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.preprocessing.text_dataset_from_directory(
    'v1_emails/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.preprocessing.text_dataset_from_directory(
    'v1_emails/test',
    batch_size=batch_size)

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

```
Found 1238 files belonging to 2 classes.
Using 991 files for training.
Found 1238 files belonging to 2 classes.
Using 247 files for validation.
Found 308 files belonging to 2 classes.
```

Looking at a few emails:

Preprocessing email text data:

In [9]:
```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

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

In [10]:
```
regex_tokenizer = nltk.RegexpTokenizer("\w+")
def text_preprocessing(content):
  content = str(content)
  content = re.sub("[^a-zA-Z]", " ", content)
  content = content.lower()
  content = content.encode("utf-8","ignore").decode()
  content = " ".join(regex_tokenizer.tokenize(content))
  for c in content:
    c.replace('\n',' ')
  words = content.split()
  stops = set(stopwords.words("english"))
  words = [w for w in words if not w in stops]

  return ' '.join(words)

train_2 = train_ds
for text_batch, label_batch in train_2:
  text_batch = text_preprocessing(text_batch)

for text_batch, label_batch in train_2.take(1):
  for i in range(10):
    print(f'Email: {text_batch.numpy()[i]}')
    label = label_batch.numpy()[i]
    print(f'Label: {label} ({class_names[label]})')
```

```
Email: b"Hi Ulf,\n\nOn Fri, 2018-04-20 at 09:35 +0200, Ulf Hansson wrote:\n\nPreviou
s multi slot implementation was removed as nobody used it and\nnobody tested it. The
re are lots of mistakes in previous implementation\nwhich are not related to request
serialization\nlike lack of slot switch / lack of adding slot id to CIU commands / e
ts...\nSo obviously it was never tested and never used at real multi slot hardwar
e.\n\n\nIn current implementation data transfers and commands to different\nhosts (s
lots) are serialized internally in the dw_mmc driver. We have\nrequest queue and whe
n .request() is called we add new request to the\nqueue. We take new request from th
e queue only if the previous one\nhas already finished.\n\nSo although hosts (slots)
have separate locks (mmc_claim|release_host())\nthe requests to different slots are
serialized by driver.\n\nIsn't that enough?\nI'm not very familiar with SD/SDIO/(e)M
MC specs so my assumptions might be wrong\nin that case please correct me.\n\n\nNeve
rtheless we had to deal somehow with existing hardware which\nhas multislot dw mmc c
ontroller and both slots are used...\n\nThis patch at least shouldn't break anything
for current users (which use\nit in single slot mode)\n\nMoreover we tested this dua
```

```
l-slot implementation and don't catch any problems\n(probably yet) except bus perfor
mance decrease in dual-slot mode (which is\nquite expected).\n\n-- \n Eugeniy Paltse
v\n"
Label: 1 (technical)
Email: b"Hi, Eduardo,\n\nOn \xe5\x9b\x9b, 2018-04-12 at 21:08 -0700, Eduardo Valenti
n wrote:\nas it is late in this merge window, I'd prefer to\n1. drop all the thermal
-soc material in the first pull request which I\nwill send out soon.\n2. you can pre
pare another pull request containing the thermal-soc\nmaterials except the exynos fi
xes\n3. exynos fixes with the problem solved can be queued for -rc2 or\nlater.\n\nth
anks,\nrui\n\n"
Label: 1 (technical)
Email: b'On Thu, Mar 15, 2018 at 01:13:07AM +0100, Maciej S. Szmigiero wrote:\n\nSur
e, it leaves the function to deal with the equiv table length only\nand the caller t
hen adds the header length. Which is actually cleaner.\n\n-- \nRegards/Gruss,\n    B
oris.\n\nGood mailing practices for 400: avoid top-posting and trim the reply.\n'
Label: 1 (technical)
Email: b"On Wed, Feb 06, 2019 at 12:14:30PM -0800, Julien Gomes wrote:\nI'm not sure
I like this.  If you have a userspace application built against\nmore recent uapi he
aders than the kernel you are actually running on, then by\ndefintion you won't have
this check in place, and you'll get EINVAL returns\nanyway.  If you just backport th
is patch to an older kernel, you'll not get the\nEINVAL return, but you will get sil
ent failures on event subscriptions that your\napplication thinks exists, but the ke
rnel doesn't recognize.  \n\nThis would make sense if you had a way to communicate b
ack to user space the\nunrecognized options, but since we don't (currently) have tha
t, I would rather\nsee the EINVAL returned than just have things not work.\n\nNeil\n
\n"
Label: 1 (technical)
Email: b'Hi,\n\nOn 05-06-18 20:18, Bob Ham wrote:\n\nYes that is fine by me and you
\'ve my permission to switch to using\njust the SPDX header.\n\nFWIW I do not believ
e the "can\'t be removed from \'this software and\nassociated documentation files (t
he "Software")\'" language\napplies to the software as a whole and not individual fi
les.\n\n\nYes you may make the same change to all files with my copyright.\n\nRegard
s,\n\nHans\n'
Label: 1 (technical)
Email: b"On Tue, Feb 05, 2019 at 03:02:23PM +0000, Robin Murphy wrote:\n\nThe bug fi
x is to handle non-vmalloc pages.  I'll see if I can do\na smaller and more bandaid-
y fix first.\n"
Label: 1 (technical)
Email: b"On Tue, Jul 03, 2018 at 05:04:10PM +1000, Andrew Jeffery wrote:\n\nI can't
take patches without any changelog text at all :(\n"
Label: 0 (nontechnical)
Email: b"Hi!\n\n\n\nYes.\n\n\nNo, I don't want that.\n\nI'd like 2 child nodes, each
specifying which HVLEDs it controls.\n\nLet me edit the original proposal.\n\n\t\t\t
\t\t\t\t\t\tPavel\n\n\n-- \n(english) http://www.livejournal.com/~pavelmachek\n(cesk
y, pictures) http://atrey.karlin.mff.cuni.cz/~pavel/picture/horses/blog.html\n\n"
Label: 1 (technical)
Email: b"On 14/01/2019 09:41, Christoph Hellwig wrote:\n\nI think the __KERNEL__ and
asm/errno.h slip-ups are things I \ncargo-culted from the arch code as a fresh-faced
noob yet to learn the \nfiner details, so ack for those parts. The forward-declarati
ons, though, \nwere a deliberate effort to minimise header dependencies and compilat
ion \nbloat for includers who absolutely wouldn't care, and specifically to \ntry to
avoid setting transitive include expectations since they always \nseem to end up bre
aking someone's config somewhere down the line. \nAdmittedly this little backwater i
s hardly comparable to the likes of \nthe sched.h business, but I'm still somewhat o
n the fence about that \nchange :/\n\nRobin.\n\n"
Label: 1 (technical)
Email: b'Interesting \xe2\x80\xa6\n\n\n\nWould you like to share any more informatio
n from this meeting?\n\n\n\nI would appreciate further indications for a correspondi
ng change acceptance.\n\nI found a feedback by Mauro Carvalho Chehab more constructi
ve.\n\n[GIT,PULL,FOR,v4.15] Cleanup fixes\nhttps://patchwork.linuxtv.org/patch/4395
7/\n\n\xe2\x80\x9c\xe2\x80\xa6\nThis time, I was nice and I took some time doing:\n
\n\t$ quilt fold < `quilt next` && quilt delete `quilt next`\n\xe2\x80\xa6\xe2\x80\x
9d\n\n\nRegards,\nMarkus\n'
Label: 0 (nontechnical)
```

Choosing a BERT model to fine-tune

In [11]:
```python
bert_model_name = 'small_bert/bert_en_uncased_L-4_H-512_A-8'
```

```
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-2/1',
    '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-4/1',
    '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-8/1',
    '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-12/1',
    '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-2/1',
    '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-4/1',
    '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-8/1',
    '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-12/1',
    '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-2/1',
    '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-4/1',
    '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-8/1',
    '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-12/1',
    '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-2/1',
    '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-4/1',
    '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-8/1',
    '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-12/1',
    '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-2/1',
    '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-4/1',
    '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-8/1',
    '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-12/1',
    '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-2/1',
    '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-4/1',
    '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-8/1',
    '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-12/1',
    '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_en_unca
sed_L-4_H-512_A-8/1
Preprocess model auto-selected: https://tfhub.dev/tensorflow/bert_en_uncased_preproc
ess/3
```

Preprocessing model

In [12]:

```
bert_preprocess_model = hub.KerasLayer(tfhub_handle_preprocess)
```

In [13]:

```
text_test = ["The driver is looking good!\n\nIt looks like you've done some kind of
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_word_ids', 'input_mask', 'input_type_ids']
Shape      : (1, 128)
Word Ids   : [ 101 1996 4062 2003 2559 2204  999 2009 3504 2066 2017 1005]
Input Mask : [1 1 1 1 1 1 1 1 1 1 1 1]
Type Ids   : [0 0 0 0 0 0 0 0 0 0 0 0]
```

Using BERT model

In [14]:

```
bert_model = hub.KerasLayer(tfhub_handle_encoder)
```

In [15]:

```
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:[ 0.8474724   0.9954415  -0.2801296   0.12758732  0.31347966
0.9054938
  0.51660323 -0.9968071  -0.056826   -0.9988778   0.1418411  -0.98870677]
Sequence Outputs Shape:(1, 128, 512)
Sequence Outputs Values:[[ 0.39939636 -0.39085585  0.9385306  ...  0.28003708  0.033
86177
  -0.40618864]
 [-0.2922875   0.40331358 -1.0200567  ... -0.57538235  0.06500234
   0.86555874]
 [-0.836157    0.07805508  0.6440214  ...  0.6109729   0.54963326
   0.5941912 ]
 ...
 [-0.3181702  -1.1716307  -1.4007791  ...  0.5933541  -0.54000527
  -0.59103113]
 [-0.40100214  0.1862419  -0.2739593  ...  0.6435037   0.38049644
```

```
  0.5307539 ]
 [-0.17033839  0.25949234  0.619225   ... -0.47313017  0.668039
  0.0198222 ]]
```

The BERT models return a map with 3 important keys: pooled_output, sequence_output, encoder_outputs:

"pooled_output" represents each input sequence as a whole. The shape is [batch_size, H]. [~ Embedding for the entire email] "sequence_output" represents each input token in the context. The shape is [batch_size, seq_length, H]. [~ contextual embedding for every token in the email] "encoder_outputs" are the intermediate activations of the L Transformer blocks. outputs["encoder_outputs"][i] is a Tensor of shape [batch_size, seq_length, 1024] with the outputs of the i-th Transformer block, for 0 <= i < L. The last value of the list is equal to sequence_output.

For the fine-tuning we use the pooled_output array.

Defining model

Fine-tuned model comprising preprocessing model + selected BERT model + 1 dense + 1 dropout layer

In [16]:
```
def build_classifier_model():
  text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
  preprocessing_layer = hub.KerasLayer(tfhub_handle_preprocess, name='preprocessing'
  encoder_inputs = preprocessing_layer(text_input)
  encoder = hub.KerasLayer(tfhub_handle_encoder, trainable=True, name='BERT_encoder'
  outputs = encoder(encoder_inputs)
  net = outputs['pooled_output']
  net = tf.keras.layers.Dropout(0.1)(net)
  net = tf.keras.layers.Dense(1, activation=None, name='classifier')(net)
  return tf.keras.Model(text_input, net)
```

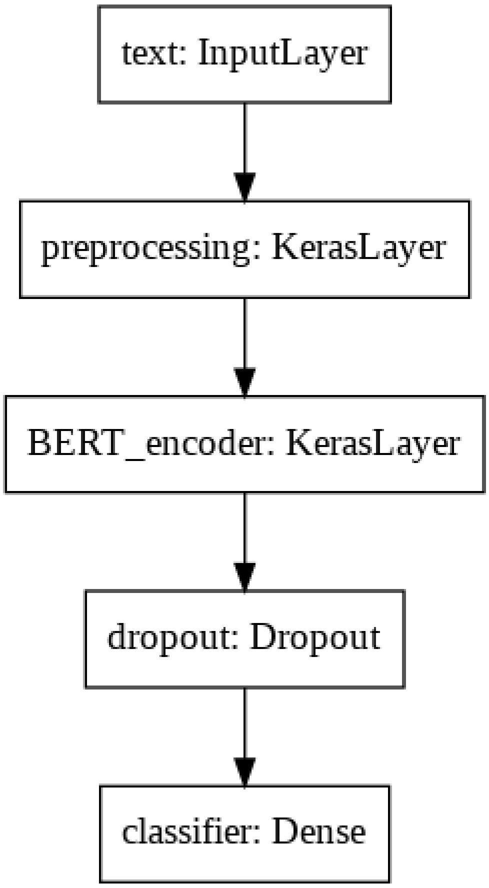
In [17]:
```
classifier_model = build_classifier_model()
bert_raw_result = classifier_model(tf.constant(text_test))
print(tf.sigmoid(bert_raw_result))
```

```
tf.Tensor([[0.66126657]], shape=(1, 1), dtype=float32)
```

Model structure

In [18]:
```
tf.keras.utils.plot_model(classifier_model)
```

Out[18]:



Model training

Loss function: binary cross entropy loss function (binary classification, model outs a probability)

In [19]:
```
loss = tf.keras.losses.BinaryCrossentropy(from_logits=True)
metrics = tf.metrics.BinaryAccuracy()
```

Optimizer: AdamW

For the learning rate (init_lr), we use the same schedule as BERT pre-training: linear decay of a notional initial learning rate, prefixed with a linear warm-up phase over the first 10% of training steps (num_warmup_steps). In line with the BERT paper, the initial learning rate is smaller for fine-tuning (best of 5e-5, 3e-5, 2e-5).

In [20]:
```
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')
```

Loading BERT model and training

In [21]:
```
classifier_model.compile(optimizer=optimizer,
                         loss=loss,
                         metrics=metrics)
```

In [22]:
```
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-51
2_A-8/1
Epoch 1/5
31/31 [==============================] - 22s 471ms/step - loss: 0.4728 - binary_accu
racy: 0.6542 - val_loss: 0.2743 - val_binary_accuracy: 0.9150
Epoch 2/5
31/31 [==============================] - 14s 442ms/step - loss: 0.3257 - binary_accu
racy: 0.8911 - val_loss: 0.2535 - val_binary_accuracy: 0.9231
Epoch 3/5
31/31 [==============================] - 14s 443ms/step - loss: 0.3018 - binary_accu
racy: 0.8985 - val_loss: 0.2684 - val_binary_accuracy: 0.9069
Epoch 4/5
31/31 [==============================] - 14s 445ms/step - loss: 0.2657 - binary_accu
racy: 0.8957 - val_loss: 0.2605 - val_binary_accuracy: 0.9150
Epoch 5/5
31/31 [==============================] - 14s 445ms/step - loss: 0.2476 - binary_accu
racy: 0.9030 - val_loss: 0.2491 - val_binary_accuracy: 0.9352
```

Evaluating model

In [23]:
```
loss, accuracy = classifier_model.evaluate(test_ds)

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

```
10/10 [==============================] - 2s 168ms/step - loss: 0.2824 - binary_accur
acy: 0.8961
Loss: 0.2824189066886902
Accuracy: 0.8961039185523987
```

Plotting accuracy, loss over time:

In [24]:
```
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)
# "bo" is for "blue dot"
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[24]:
```
<matplotlib.legend.Legend at 0x7f97af55bed0>
```

In [34]:

```
# testing:
examples = ["Was looking for that. Thanks. Speaking of that, recent lksctp-tools \
got some defines to help knowing which features the available kernel headers \
have as it now probes if specific struct members are available or not. \
Though yeah, it also wouldn't help in this case, just mentioning it.",
"Sorry but I don't like imposing a run-time check on everybody \
when stack-based requests are the odd ones out.  If we're going to make \
this a run-time check (I'd much prefer a compile-time check, but I \
understand that this may involve too much churn), then please do it \
for stack-based request users only.", "And I was just reminded about huge \
pages. But still, my point of finding a compromise still stands.", \
"Since when is the cover letter \
mandatory? I understand that is helps for a complicated patch set \
to explain the problem and solution in the cover letter, but for this \
simple test case addition what's the point? And there is nothing \
forcing a cover letter in", "I'm not exactly sure how Linux switch driver \
works, but from DT perspective I think we should rather have \
*hardware* described instead of a common Linux case. If I'm right, \
we should rather have all 3 switch ports described (5, 7,8) and have \
Linux just use the one it needs."]
# technical, non-technical, technical, non-technical, technical

def print_results(inputs, results):
  for i in range(len(inputs)):
    prediction = "Non-technical"
    if results[i][0]>=0.5:
      prediction = "Technical"
    print("Input:", inputs[i], "\nScore:", results[i][0], "\nPrediction:",prediction

results = tf.sigmoid(classifier_model(tf.constant(examples)))
print_results(examples, results)
```

```
Input: Was looking for that. Thanks. Speaking of that, recent lksctp-tools got some
defines to help knowing which features the available kernel headers have as it now p
robes if specific struct members are available or not. Though yeah, it also wouldn't
help in this case, just mentioning it.
Score: tf.Tensor(0.9106656, shape=(), dtype=float32)
Prediction: Technical
Input: Sorry but I don't like imposing a run-time check on everybody when stack-base
d requests are the odd ones out.  If we're going to make this a run-time check (I'd
much prefer a compile-time check, but I understand that this may involve too much ch
```

```
urn), then please do it for stack-based request users only.
Score: tf.Tensor(0.7932073, shape=(), dtype=float32)
Prediction: Technical
Input: And I was just reminded about huge pages. But still, my point of finding a co
mpromise still stands.
Score: tf.Tensor(0.89355475, shape=(), dtype=float32)
Prediction: Technical
Input: Since when is the cover letter mandatory? I understand that is helps for a co
mplicated patch set to explain the problem and solution in the cover letter, but for
this simple test case addition what's the point? And there is nothing forcing a cove
r letter in
Score: tf.Tensor(0.8910215, shape=(), dtype=float32)
Prediction: Technical
Input: I'm not exactly sure how Linux switch driver works, but from DT perspective I
think we should rather have *hardware* described instead of a common Linux case. If
I'm right, we should rather have all 3 switch ports described (5, 7,8) and have Linu
x just use the one it needs.
Score: tf.Tensor(0.96671313, shape=(), dtype=float32)
Prediction: Technical
```