Setup

In [1]:
```
# A dependency of the preprocessing for BERT inputs
!pip install -q tensorflow-text
```

```
     |████████████████████████████████| 3.4MB 2.8MB/s
```

In [3]:
```
# Using AdamW optimizer
!pip install -q tf-models-official==2.4
```

```
     |████████████████████████████████| 1.1MB 2.8MB/s
     |████████████████████████████████| 686kB 23.6MB/s
     |████████████████████████████████| 174kB 16.5MB/s
     |████████████████████████████████| 1.2MB 17.7MB/s
     |████████████████████████████████| 102kB 7.6MB/s
     |████████████████████████████████| 645kB 24.0MB/s
     |████████████████████████████████| 358kB 24.0MB/s
     |████████████████████████████████| 51kB 5.2MB/s
     |████████████████████████████████| 38.2MB 65kB/s
  Building wheel for py-cpuinfo (setup.py) ... done
  Building wheel for seqeval (setup.py) ... done
```

In [4]:
```
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 [5]:
```
url = 'https://github.com/ahlraf/point/blob/main/civ_uc.tar.gz?raw=true'
dataset = tf.keras.utils.get_file('civ_uc.tar.gz', url,
                                  untar=True, cache_dir='.',
                                  cache_subdir='')
```

```
Downloading data from https://github.com/ahlraf/point/blob/main/civ_uc.tar.gz?raw=tr
ue
73728/68315 [================================] - 0s 0us/step
```

In [6]:
```
dataset_dir = os.path.join(os.path.dirname(dataset), 'civ_uc')
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(
    'civ_uc/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(
    'civ_uc/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(
    'civ_uc/test',
    batch_size=batch_size)

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

```
Found 135 files belonging to 2 classes.
Using 108 files for training.
Found 135 files belonging to 2 classes.
Using 27 files for validation.
Found 33 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"On 13/03/18 18:18, Ard Biesheuvel wrote:\n\nThis is indeed guaranteed. For
FTRACE use case. If it's being called from FTRACE in\nrun time, this would mean ther
e were long calls in this module section, which in\nturn means, get_module_plt() was
called at least once for this module and this\nsection.\n\nThis doesn't hold in gene
ral, though.\n\nIn any case, if you insist, I can try to rework the whole stuff impl
ementing module_finalize().\n\n-- \nBest regards,\nAlexander Sverdlin.\n"
Label: 0 (civil)
Email: b'Are you going to answer any of my remaining questions in a more constructiv
e way?\n\nRegards,\nMarkus\n'
Label: 1 (uncivil)
Email: b'Michal Hocko wrote:\n\nThen, I am wondering why we are holding mmap_sem whe
n calling\nmigrate_pages() in existing code.\nhttp://elixir.free-electrons.com/linu
x/latest/source/mm/migrate.c#L1576\n\n\nSorry, I missed that. If mmap_sem is not nee
ded for migrate_pages(),\nplease ignore this patch.\n\n\n\n-- \nBest Regards,\nYan Z
i\n\n\n'
```

Label: 0 (civil)
Email: b"IMO symlinks are mostly ending in a mess, URLs are never stable.\nThere is a \n\n https://www.kernel.org/doc/html/latest/objects.inv\n\nto handle such requirements. Take a look at *intersphinx* :\n\n http://www.sphinx-doc.org/en/stable/ext/intersphinx.html\n\nto see how it works: Each Sphinx HTML build creates a file named objects.inv that\ncontains a mapping from object names to URIs relative to the HTML set\xe2\x80\x99s root.\n\nThis means articles from external (like lwn articles) has to be recompiled.\nNot perfect, but a first solution. \n\n\nI really like them, factually valuable comments .. please\nexpress your concern so that we have a chance to move on.\n\n\nI think that's a pity.\n\n-- Markus --\n"
Label: 1 (uncivil)
Email: b"On Wed, 19 Sep 2018, Tvrtko Ursulin wrote:\n\nIt would be very helpful if you cc all involved people on the cover letter\ninstead of just cc'ing your own pile of email addresses. CC'ed now.\n\n\nThis is really not helpful. The cover letter and the change logs should\ncontain a summary of that discussion and a proper justification of the\nproposed change. Just saying 'sysadmins might want to allow' is not useful\nat all, it's yet another 'I want a pony' thing.\n\nI read through the previous thread and there was a clear request to involve\nsecurity people into this. Especially those who are deeply involved with\nhardware side channels. I don't see anyone Cc'ed on the whole series.\n\nFor the record, I'm not buying the handwavy 'more noise' argument at\nall. It wants a proper analysis and we need to come up with criteria which\nPMUs can be exposed at all.\n\nAll of this want's a proper documentation clearly explaining the risks and\nscope of these knobs per PMU. Just throwing magic knobs at sysadmins and\nthen saying 'its their problem to figure it out' is not acceptable.\n\nThanks,\n\n\ttglx\n"
Label: 1 (uncivil)
Email: b'Thank you so much for many style, formatting and other issues fixes and also for\nintegration of \'check_at_most_once\' patch, it saved me several review iterations.\n\n\nRegarding free of sg in two error paths, you were correct.\nI fixed it by placing several error labels to differentiate each handling.\nI also noted that reqdata_arr[b].req was not released properly, this is also fixed.\nfollowing is a diff of my fix based on your modifications.\n(I can send it in a patch format, but it doesn\'t include a fix for Eric Biggers comments)\n\n\n@@ -573,10 +573,9 @@ static void verity_verify_io(struct dm_verity_io *io)\n   verity_bv_skip_block(v, io, &io->iter);\n   continue;\n   }\n-\n   reqdata_arr[b].req = ahash_request_alloc(v->tfm, GFP_NOIO);\n   if (unlikely(reqdata_arr[b].req == NULL))\n-   goto err_memfree;\n+   goto err_mem_req;\n   ahash_request_set_tfm(reqdata_arr[b].req, v->tfm);\n \n   /* +1 for the salt buffer */\n@@ -586,7 +585,7 @@ static void verity_verify_io(struct dm_verity_io *io)\n   GFP_NOIO);\n   if (!sg) {\n   DMERR_LIMIT("%s: kmalloc_array failed", __func__);\n-   goto err_memfree;\n+   goto err_mem_sg;\n   }\n   sg_init_table(sg, num_of_buffs);\n   // FIXME: if we \'err_memfree\' (or continue;) below how does this sg get kfree()\'d?\n@@ -595,7 +594,7 @@ static void verity_verify_io(struct dm_verity_io *io)\n   reqdata_arr[b].want_digest,\n   &reqdata_arr[b].fec_io, &is_zero);\n   if (unlikely(r < 0))\n-   goto err_memfree;\n+   goto err_mem;\n \n   if (is_zero) {\n   /*\n@@ -605,7 +604,7 @@ static void verity_verify_io(struct dm_verity_io *io)\n   r = verity_for_bv_block(v, io, &io->iter,\n   verity_bv_zero);\n   if (unlikely(r < 0))\n-   goto err_memfree;\n+   goto err_mem;\n   verity_cb_complete(iodata, r);\n   continue;\n   }\n@@ -644,7 +643,11 @@ static void verity_verify_io(struct dm_verity_io *io)\n   }\n   return;\n \n-err_memfree:\n+err_mem:\n   kfree(sg);\n+err_mem_sg:\n+   ahash_request_free(reqdata_arr[b].req);\n+err_mem_req:\n   /*\n   * reduce expected requests by the number of unsent\n   * requests, -1 accounting for the current block\n   */\n   atomic_sub(blocks - b - 1, &iodata->expected_reqs);\n   verity_cb_complete(iodata, -EIO);\n\n\nI took your modifications and working upon it. \n\n'
Label: 0 (civil)
Email: b'Hi!\n\n\nThis is better than my proposal. Thanks!\n\t\t\t\t\t\t\t\tPavel\n-- \n(english) http://www.livejournal.com/~pavelmachek\n(cesky, pictures) http://atrey.karlin.mff.cuni.cz/~pavel/picture/horses/blog.html\n'
Label: 0 (civil)
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 (civil)

```
Email: b"Ah only if google could simply answer all our questions!\n\n\nIt's not like
there is or isn't a security risk and that you\ncan say that it is or it isn't in a
global way.\n\nEssentially these are channels of information. The channels always ex
ist\nin form of timing variances for any shared resource (like shared caches\nor sha
red memory/IO/interconnect bandwidth) that can be measured.\n\nPerfmon counters make
the channels generally less noisy, but they do not cause\nthem.\n\nTo really close t
hem completely you would need to avoid sharing\nanything, or not allowing to measure
time, neither of which is practical\nshort of an air gap.\n\nThere are reasonable as
sesments you can make either way and the answers\nwill be different based on your re
quirements. There isn't a single\nanswer that works for everyone. \n\nThere are case
s where it isn't a problem at all.\n\nIf you don't have multiple users on the system
your tolerance\nshould be extremely high.\n\nFor users who have multiple users there
can be different tradeoffs.\n\nSo there isn't a single answer, and that is why it is
important\nthat this if configurable.\n\n-Andi\n"
Label: 1 (uncivil)
Email: b"On Tue, 30 Jan 2018 17:14:45 +0200\nIgor Stoppa <igor.stoppa@huawei.com> wr
ote:\n\n\nPlease don't put plain-text files into core-api - that's a directory full
\nof RST documents.  Your document is 99.9% RST already, better to just\nfinish the
job and tie it into the rest of the kernel docs.\n\n\nWe might as well put the SPDX
tag here, it's a new file.\n\n\nThis is all good information, but I'd suggest it bel
ongs more in the 0/n\npatch posting than here.  The introduction of *this* document
should say\nwhat it actually covers.\n\n\nThis seems like a relevant and important a
spect of the API that shouldn't\nbe buried in the middle of a section talking about
random things.\n\n\nSo one gets this far, but has no actual idea of how to do these
things.\nWhich leads me to wonder: what is this document for?  Who are you expecting
\nto read it?\n\nYou could improve things a lot by (once again) going to RST and usi
ng\ndirectives to bring in the kerneldoc comments from the source (which, I\nnote, d
o exist).  But I'd suggest rethinking this document and its\naudience.  Most of the
people reading it are likely wanting to learn how to\n*use* this API; I think it wou
ld be best to not leave them frustrated.\n\nThanks,\n\njon\n"
Label: 1 (uncivil)
```

Choosing a BERT model to fine-tune

In [12]:

```
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 [13]:
```
bert_preprocess_model = hub.KerasLayer(tfhub_handle_preprocess)
```

In [14]:
```
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_type_ids', 'input_word_ids', 'input_mask']
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 [15]:
```
bert_model = hub.KerasLayer(tfhub_handle_encoder)
```

In [16]:
```
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:[ 0.8474725   0.99544144 -0.28012967  0.12758812  0.3134793
0.9054936
  0.5166035  -0.99680704 -0.056826   -0.998878    0.14184158 -0.98870665]
Sequence Outputs Shape:(1, 128, 512)
Sequence Outputs Values:[[ 0.39939716 -0.39085555  0.93853164 ...  0.2800367   0.033
86121
  -0.4061886 ]
 [-0.29228777  0.40331376 -1.0200582  ... -0.5753827   0.06500192
   0.86555845]
 [-0.8361559   0.07805473  0.6440225  ...  0.6109724   0.5496335
   0.594189  ]
 ...
 [-0.31816992 -1.171632   -1.4007776  ...  0.5933536  -0.5400041
  -0.59102964]
 [-0.40100315  0.18624073 -0.27396095 ...  0.64350426  0.38049635
   0.53075486]
 [-0.1703408   0.2594934   0.61922497 ... -0.47313112  0.66804034
   0.01982282]]
```

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 [17]:
```
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)
```

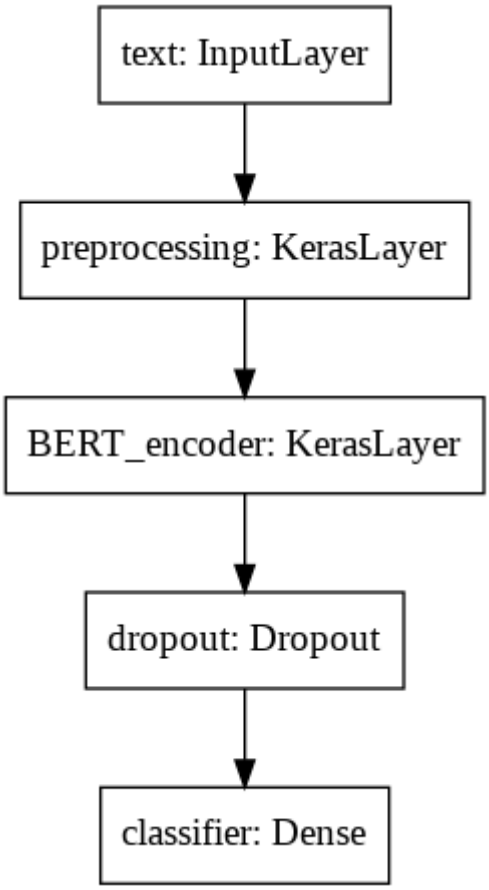
In [18]:
```
classifier_model = build_classifier_model()
bert_raw_result = classifier_model(tf.constant(text_test))
print(tf.sigmoid(bert_raw_result))
```

```
tf.Tensor([[0.6166136]], shape=(1, 1), dtype=float32)
```

Model structure

In [19]:
```
tf.keras.utils.plot_model(classifier_model)
```

Out[19]:



Model training

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

In [20]:
```
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 [21]:
```
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 [22]:
```
classifier_model.compile(optimizer=optimizer,
                         loss=loss,
                         metrics=metrics)
```

In [23]:
```
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
4/4 [==============================] - 8s 371ms/step - loss: 0.7775 - binary_accurac
y: 0.3360 - val_loss: 0.5965 - val_binary_accuracy: 0.5556
Epoch 2/5
4/4 [==============================] - 1s 187ms/step - loss: 0.6335 - binary_accurac
y: 0.6766 - val_loss: 0.6032 - val_binary_accuracy: 0.6296
Epoch 3/5
4/4 [==============================] - 1s 188ms/step - loss: 0.5997 - binary_accurac
y: 0.6706 - val_loss: 0.6094 - val_binary_accuracy: 0.5185
Epoch 4/5
4/4 [==============================] - 1s 192ms/step - loss: 0.5396 - binary_accurac
y: 0.7222 - val_loss: 0.6120 - val_binary_accuracy: 0.6296
Epoch 5/5
4/4 [==============================] - 1s 194ms/step - loss: 0.5317 - binary_accurac
y: 0.6794 - val_loss: 0.6092 - val_binary_accuracy: 0.5926
```

Evaluating model

In [24]:
```
loss, accuracy = classifier_model.evaluate(test_ds)

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

```
2/2 [==============================] - 0s 13ms/step - loss: 0.5826 - binary_accurac
y: 0.6667
Loss: 0.5826314687728882
Accuracy: 0.6666666865348816
```

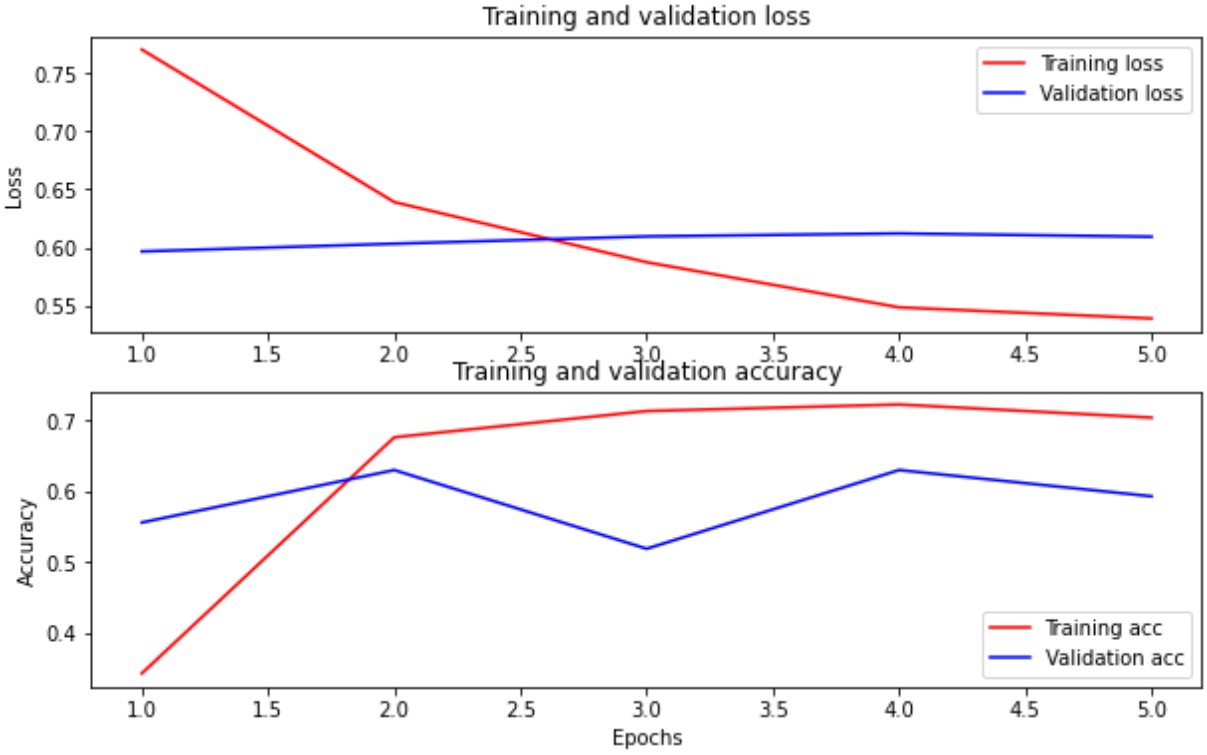
Plotting accuracy, loss over time:

In [25]:
```
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[25]:
```
<matplotlib.legend.Legend at 0x7f0372062f50>
```



In [27]:
```
# testing:
examples = ["I would really like to get an ack from the people who have been \
deep into this first.  If you can get that, and preferably resubmit with a \
less condescending changelog, I can pick it up.", "I like the idea and I think \
it's good direction to go, but could you please share some from perf stat or \
whatever you used to meassure the new performance?", "What's advertisement \
there? Huch? Care to tell what's a lie instead of making bold statements? \
Thanks, tglx", "There probably is a decent compromise to find between \
'not accepting a single additional byte' and accepting several GB. \
For example how likely is it that the growth of this structure make it \
go over a page? I would hope not at all. By choosing a large but decent \
high limit, I think we can find a future-compatible compromise that doesn't \
rely on a preliminary getsockopt() just for structure trucation decision..."]

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

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

```
Input: I would really like to get an ack from the people who have been deep into thi
s first.  If you can get that, and preferably resubmit with a less condescending cha
```

```
ngelog, I can pick it up.
Score: tf.Tensor(0.75841796, shape=(), dtype=float32)
Prediction: Civil
Input: I like the idea and I think it's good direction to go, but could you please s
hare some from perf stat or whatever you used to meassure the new performance?
Score: tf.Tensor(0.8285885, shape=(), dtype=float32)
Prediction: Civil
Input: What's advertisement there? Huch? Care to tell what's a lie instead of making
bold statements? Thanks, tglx
Score: tf.Tensor(0.6883302, shape=(), dtype=float32)
Prediction: Civil
Input: There probably is a decent compromise to find between 'not accepting a single
additional byte' and accepting several GB. For example how likely is it that the gro
wth of this structure make it go over a page? I would hope not at all. By choosing a
large but decent high limit, I think we can find a future-compatible compromise that
doesn't rely on a preliminary getsockopt() just for structure trucation decision...
Score: tf.Tensor(0.7940565, shape=(), dtype=float32)
Prediction: Civil
```