Using pre-trained embedding with Tensorflow Hub

Learning Objectives

1. How to instantiate a Tensorflow Hub module
2. How to find pretrained Tensorflow Hub module for variety of purposes
3. How to use a pre-trained TF Hub text modules to generate sentence vectors
4. How to incorporate a pre-trained TF-Hub module into a Keras model

Introduction

In this notebook, we will implement text models to recognize the probable source (Github, Tech-Crunch, or The New-York Times) of the titles we have in the title dataset.

First, we will load and pre-process the texts and labels so that they are suitable to be fed to sequential Keras models with first layer being TF-hub pre-trained modules. Thanks to this first layer, we won't need to tokenize and integerize the text before passing it to our models. The pre-trained layer will take care of that for us, and consume directly raw text. However, we will still have to one-hot-encode each of the 3 classes into a 3 dimensional basis vector.

Then we will build, train and compare simple models starting with different pre-trained TF-Hub layers.

In [1]:
```
!sudo chown -R jupyter:jupyter /home/jupyter/training-data-analyst
```

In [2]:
```
!pip install --user google-cloud-bigquery==1.25.0
```

```
Collecting google-cloud-bigquery==1.25.0
  Downloading google_cloud_bigquery-1.25.0-py2.py3-none-any.whl (169 kB)
     |████████████████████████████████| 169 kB 8.3 MB/s
Requirement already satisfied: protobuf>=3.6.0 in /opt/conda/lib/python3.7/site-
packages (from google-cloud-bigquery==1.25.0) (3.18.1)
Collecting google-resumable-media<0.6dev,>=0.5.0
  Downloading google_resumable_media-0.5.1-py2.py3-none-any.whl (38 kB)
Collecting google-auth<2.0dev,>=1.9.0
  Downloading google_auth-1.35.0-py2.py3-none-any.whl (152 kB)
     |████████████████████████████████| 152 kB 58.7 MB/s
Collecting google-cloud-core<2.0dev,>=1.1.0
  Downloading google_cloud_core-1.7.2-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: six<2.0.0dev,>=1.13.0 in /opt/conda/lib/python3.
7/site-packages (from google-cloud-bigquery==1.25.0) (1.16.0)
Collecting google-api-core<2.0dev,>=1.15.0
  Downloading google_api_core-1.31.3-py2.py3-none-any.whl (93 kB)
     |████████████████████████████████| 93 kB 2.1 MB/s
Requirement already satisfied: pytz in /opt/conda/lib/python3.7/site-packages (f
rom google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery==1.25.0) (2021.3)
Requirement already satisfied: packaging>=14.3 in /opt/conda/lib/python3.7/site-
```

```
packages (from google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery==1.25.0)
(21.0)
Requirement already satisfied: setuptools>=40.3.0 in /opt/conda/lib/python3.7/si
te-packages (from google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery==1.25.
0) (58.2.0)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /opt/conda/lib/pyth
on3.7/site-packages (from google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery
==1.25.0) (2.25.1)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in /opt/c
onda/lib/python3.7/site-packages (from google-api-core<2.0dev,>=1.15.0->google-c
loud-bigquery==1.25.0) (1.53.0)
Collecting protobuf>=3.6.0
  Downloading protobuf-3.17.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.
whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 64.9 MB/s
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.7/site-pa
ckages (from google-auth<2.0dev,>=1.9.0->google-cloud-bigquery==1.25.0) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /opt/conda/lib/python3.
7/site-packages (from google-auth<2.0dev,>=1.9.0->google-cloud-bigquery==1.25.0)
(4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.
7/site-packages (from google-auth<2.0dev,>=1.9.0->google-cloud-bigquery==1.25.0)
(0.2.7)
Requirement already satisfied: pyparsing>=2.0.2 in /opt/conda/lib/python3.7/site
-packages (from packaging>=14.3->google-api-core<2.0dev,>=1.15.0->google-cloud-b
igquery==1.25.0) (2.4.7)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /opt/conda/lib/python3.7/
site-packages (from pyasn1-modules>=0.2.1->google-auth<2.0dev,>=1.9.0->google-cl
oud-bigquery==1.25.0) (0.4.8)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.7/sit
e-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.0->go
ogle-cloud-bigquery==1.25.0) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/si
te-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.0->g
oogle-cloud-bigquery==1.25.0) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.
7/site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.
0->google-cloud-bigquery==1.25.0) (1.26.7)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-pac
kages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.0->google-
cloud-bigquery==1.25.0) (2.10)
Installing collected packages: protobuf, google-auth, google-api-core, google-re
sumable-media, google-cloud-core, google-cloud-bigquery
ERROR: pip's dependency resolver does not currently take into account all the pa
ckages that are installed. This behaviour is the source of the following depende
ncy conflicts.
tensorflow-io 0.18.0 requires tensorflow-io-gcs-filesystem==0.18.0, which is not
installed.
explainable-ai-sdk 1.3.2 requires xai-image-widget, which is not installed.
tfx-bsl 1.3.0 requires absl-py<0.13,>=0.9, but you have absl-py 0.15.0 which is
 incompatible.
tfx-bsl 1.3.0 requires google-api-python-client<2,>=1.7.11, but you have google-
api-python-client 2.27.0 which is incompatible.
tfx-bsl 1.3.0 requires pyarrow<3,>=1, but you have pyarrow 5.0.0 which is incomp
atible.
tensorflow 2.6.0 requires six~=1.15.0, but you have six 1.16.0 which is incompat
ible.
tensorflow 2.6.0 requires tensorboard~=2.6, but you have tensorboard 2.5.0 which
is incompatible.
tensorflow 2.6.0 requires typing-extensions~=3.7.4, but you have typing-extensio
ns 3.10.0.2 which is incompatible.
```

```
tensorflow 2.6.0 requires wrapt~=1.12.1, but you have wrapt 1.13.2 which is inco
mpatible.
tensorflow-transform 1.3.0 requires absl-py<0.13,>=0.9, but you have absl-py 0.1
5.0 which is incompatible.
tensorflow-transform 1.3.0 requires pyarrow<3,>=1, but you have pyarrow 5.0.0 wh
ich is incompatible.
tensorflow-metadata 1.2.0 requires absl-py<0.13,>=0.9, but you have absl-py 0.1
5.0 which is incompatible.
tensorflow-io 0.18.0 requires tensorflow<2.6.0,>=2.5.0, but you have tensorflow
 2.6.0 which is incompatible.
google-cloud-storage 1.42.3 requires google-resumable-media<3.0dev,>=1.3.0; pyth
on_version >= "3.6", but you have google-resumable-media 0.5.1 which is incompat
ible.
cloud-tpu-client 0.10 requires google-api-python-client==1.8.0, but you have goo
gle-api-python-client 2.27.0 which is incompatible.
apache-beam 2.33.0 requires dill<0.3.2,>=0.3.1.1, but you have dill 0.3.4 which
 is incompatible.
apache-beam 2.33.0 requires httplib2<0.20.0,>=0.8, but you have httplib2 0.20.1
 which is incompatible.
apache-beam 2.33.0 requires pyarrow<5.0.0,>=0.15.1, but you have pyarrow 5.0.0 w
hich is incompatible.
Successfully installed google-api-core-1.31.3 google-auth-1.35.0 google-cloud-bi
gquery-1.25.0 google-cloud-core-1.7.2 google-resumable-media-0.5.1 protobuf-3.1
7.3
```

Note: Restart your kernel to use updated packages.

Kindly ignore the deprecation warnings and incompatibility errors related to google-cloud-storage.

In [1]:
```
import os

from google.cloud import bigquery
import pandas as pd
```

In [2]:
```
%load_ext google.cloud.bigquery
```

```
The google.cloud.bigquery extension is already loaded. To reload it, use:
  %reload_ext google.cloud.bigquery
```

Replace the variable values in the cell below:

In [3]:
```
PROJECT = "qwiklabs-gcp-01-7f8910be06fa"  # Replace with your PROJECT
BUCKET = PROJECT  # defaults to PROJECT
REGION = "us-central1"  # Replace with your REGION
SEED = 0
```

Create a Dataset from BigQuery

Hacker news headlines are available as a BigQuery public dataset. The dataset contains all headlines from the sites inception in October 2006 until October 2015.

Here is a sample of the dataset:

In [4]:
```
%%bigquery --project $PROJECT
```

```
SELECT
    url, title, score
FROM
    `bigquery-public-data.hacker_news.stories`
WHERE
    LENGTH(title) > 10
    AND score > 10
    AND LENGTH(url) > 0
LIMIT 10
```

Out[4]:

	url	title	score
0	http://www.dumpert.nl/mediabase/6560049/3eb18e...	Calling the NSA: "I accidentally deleted an e-...	258
1	http://blog.liip.ch/archive/2013/10/28/hhvm-an...	Amazing performance with HHVM and PHP with a S...	11
2	http://www.gamedev.net/page/resources/_/techni...	A Journey Through the CPU Pipeline	11
3	http://jfarcand.wordpress.com/2011/02/25/atmos...	Atmosphere Framework 0.7 released: GWT, Wicket...	11
4	http://tech.gilt.com/post/90578399884/immutabl...	Immutable Infrastructure with Docker and EC2 [...	11
5	http://thechangelog.com/post/501053444/episode...	Changelog 0.2.0 - node.js w/Felix Geisendorfer	11
6	http://openangelforum.com/2010/09/09/second-bo...	Second Open Angel Forum in Boston Oct 13th--fr...	11
7	http://bredele.github.io/async	A collection of JavaScript asynchronous patterns	11
8	http://www.smashingmagazine.com/2007/08/25/20-...	20 Free and Fresh Icon Sets	11
9	http://www.cio.com/article/147801/Study_Finds_...	Study: Only 1 in 5 Workers is "Engaged" in The...	11

Let's do some regular expression parsing in BigQuery to get the source of the newspaper article from the URL.

In [5]:

```
%%bigquery --project $PROJECT

SELECT
    ARRAY_REVERSE(SPLIT(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.'))[OFFSET(1)]
    COUNT(title) AS num_articles
FROM
    `bigquery-public-data.hacker_news.stories`
WHERE
    REGEXP_CONTAINS(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.com$')
    AND LENGTH(title) > 10
GROUP BY
    source
ORDER BY num_articles DESC
  LIMIT 100
```

Out[5]:

	source	num_articles

	source	num_articles
0	blogspot	41386
1	github	36525
2	techcrunch	30891
3	youtube	30848
4	nytimes	28787
...	...	...
95	f5	1254
96	gamasutra	1249
97	cnbc	1229
98	indiatimes	1223
99	computerworlduk	1166

100 rows × 2 columns

Now that we have good parsing of the URL to get the source, let's put together a dataset of source and titles. This will be our labeled dataset for machine learning.

In [6]:
```
regex = '.*://(.[^/]+)/'


sub_query = """
SELECT
    title,
    ARRAY_REVERSE(SPLIT(REGEXP_EXTRACT(url, '{0}'), '.'))[OFFSET(1)] AS source

FROM
    `bigquery-public-data.hacker_news.stories`
WHERE
    REGEXP_CONTAINS(REGEXP_EXTRACT(url, '{0}'), '.com$')
    AND LENGTH(title) > 10
""".format(regex)


query = """
SELECT
    LOWER(REGEXP_REPLACE(title, '[^a-zA-Z0-9 $.-]', ' ')) AS title,
    source
FROM
  ({sub_query})
WHERE (source = 'github' OR source = 'nytimes' OR source = 'techcrunch')
""".format(sub_query=sub_query)

print(query)
```

```
SELECT
    LOWER(REGEXP_REPLACE(title, '[^a-zA-Z0-9 $.-]', ' ')) AS title,
    source
FROM
  (
SELECT
```

```
    title,
    ARRAY_REVERSE(SPLIT(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.'))[OFFSET(1)]
AS source

FROM
    `bigquery-public-data.hacker_news.stories`
WHERE
    REGEXP_CONTAINS(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.com$')
    AND LENGTH(title) > 10
)
WHERE (source = 'github' OR source = 'nytimes' OR source = 'techcrunch')
```

For ML training, we usually need to split our dataset into training and evaluation datasets (and perhaps an independent test dataset if we are going to do model or feature selection based on the evaluation dataset). AutoML however figures out on its own how to create these splits, so we won't need to do that here.

In [7]:
```
bq = bigquery.Client(project=PROJECT)
title_dataset = bq.query(query).to_dataframe()
title_dataset.head()
```

Out[7]:

	title	source
0	this guy just found out how to bypass adblocker	github
1	show hn dodo command line task management f...	github
2	without coding test test automation for javas...	github
3	clojure s first code commit authored 8 years ...	github
4	hikaricp a solid high-performance jdbc connect...	github

AutoML for text classification requires that

- the dataset be in csv form with
- the first column being the texts to classify or a GCS path to the text
- the last column to be the text labels

The dataset we pulled from BiqQuery satisfies these requirements.

In [8]:
```
print("The full dataset contains {n} titles".format(n=len(title_dataset)))
```

```
The full dataset contains 96203 titles
```

Let's make sure we have roughly the same number of labels for each of our three labels:

In [9]:
```
title_dataset.source.value_counts()
```

Out[9]:
```
github        36525
techcrunch    30891
nytimes       28787
Name: source, dtype: int64
```

Finally we will save our data, which is currently in-memory, to disk.

We will create a csv file containing the full dataset and another containing only 1000 articles for development.

Note: It may take a long time to train AutoML on the full dataset, so we recommend to use the sample dataset for the purpose of learning the tool.

In [10]:
```
DATADIR = './data/'

if not os.path.exists(DATADIR):
    os.makedirs(DATADIR)
```

In [11]:
```
FULL_DATASET_NAME = 'titles_full.csv'
FULL_DATASET_PATH = os.path.join(DATADIR, FULL_DATASET_NAME)

# Let's shuffle the data before writing it to disk.
title_dataset = title_dataset.sample(n=len(title_dataset))

title_dataset.to_csv(
    FULL_DATASET_PATH, header=False, index=False, encoding='utf-8')
```

Now let's sample 1000 articles from the full dataset and make sure we have enough examples for each label in our sample dataset (see here for further details on how to prepare data for AutoML).

In [12]:
```
sample_title_dataset = title_dataset.sample(n=1000)
sample_title_dataset.source.value_counts()
```

Out[12]:
```
github         388
techcrunch     310
nytimes        302
Name: source, dtype: int64
```

Let's write the sample datatset to disk.

In [13]:
```
SAMPLE_DATASET_NAME = 'titles_sample.csv'
SAMPLE_DATASET_PATH = os.path.join(DATADIR, SAMPLE_DATASET_NAME)

sample_title_dataset.to_csv(
    SAMPLE_DATASET_PATH, header=False, index=False, encoding='utf-8')
```

In [14]:
```
sample_title_dataset.head()
```

Out[14]:

	title	source
80248	new mac pro uses 74 less aluminum 68 less p...	techcrunch
73501	a deeper look inside apple s secrecy and susta...	techcrunch
77014	the real video twitter 12seconds.tv 500 alph...	techcrunch
37081	a ray of hope on climate change	nytimes
94070	google debuts its first apple watch app with ...	techcrunch

In [15]:
```
import datetime
import os
import shutil

import pandas as pd
import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard, EarlyStopping
from tensorflow_hub import KerasLayer
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical


print(tf.__version__)
```

```
2.6.0
```

In [16]:
```
%matplotlib inline
```

Let's start by specifying where the information about the trained models will be saved as well as where our dataset is located:

In [17]:
```
MODEL_DIR = "./text_models"
DATA_DIR = "./data"
```

Loading the dataset

As in the previous labs, our dataset consists of titles of articles along with the label indicating from which source these articles have been taken from (GitHub, Tech-Crunch, or the New-York Times):

In [18]:
```
ls ./data/
```

```
titles_full.csv  titles_sample.csv
```

In [19]:
```
DATASET_NAME = "titles_full.csv"
TITLE_SAMPLE_PATH = os.path.join(DATA_DIR, DATASET_NAME)
COLUMNS = ['title', 'source']

titles_df = pd.read_csv(TITLE_SAMPLE_PATH, header=None, names=COLUMNS)
titles_df.head()
```

Out[19]:

	title	source
0	clj3d a clojure graphic library	github
1	factory girl for scala	github
2	searching for meaningful markers of aging	nytimes
3	how technology wrecks the middle class	nytimes
4	chris anderson s free is now available for fre...	techcrunch

Let's look again at the number of examples per label to make sure we have a well-balanced dataset:

In [20]:
```
titles_df.source.value_counts()
```

Out[20]:
```
github         36525
techcrunch     30891
nytimes        28787
Name: source, dtype: int64
```

Preparing the labels

In this lab, we will use pre-trained TF-Hub embeddings modules for english for the first layer of our models. One immediate advantage of doing so is that the TF-Hub embedding module will take care for us of processing the raw text. This also means that our model will be able to consume text directly instead of sequences of integers representing the words.

However, as before, we still need to preprocess the labels into one-hot-encoded vectors:

In [21]:
```
CLASSES = {
    'github': 0,
    'nytimes': 1,
    'techcrunch': 2
}
N_CLASSES = len(CLASSES)
```

In [22]:
```
def encode_labels(sources):
    classes = [CLASSES[source] for source in sources]
    one_hots = to_categorical(classes, num_classes=N_CLASSES)
    return one_hots
```

In [23]:
```
encode_labels(titles_df.source[:4])
```

Out[23]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.]], dtype=float32)
```

Preparing the train/test splits

Let's split our data into train and test splits:

In [24]:
```
N_TRAIN = int(len(titles_df) * 0.95)

titles_train, sources_train = (
    titles_df.title[:N_TRAIN], titles_df.source[:N_TRAIN])

titles_valid, sources_valid = (
    titles_df.title[N_TRAIN:], titles_df.source[N_TRAIN:])
```

To be on the safe side, we verify that the train and test splits have roughly the same number of

examples per class.

Since it is the case, accuracy will be a good metric to use to measure the performance of our models.

In [25]:
```
sources_train.value_counts()
```

Out[25]:
```
github        34699
techcrunch    29343
nytimes       27350
Name: source, dtype: int64
```

In [26]:
```
sources_valid.value_counts()
```

Out[26]:
```
github        1826
techcrunch    1548
nytimes       1437
Name: source, dtype: int64
```

Now let's create the features and labels we will feed our models with:

In [27]:
```
X_train, Y_train = titles_train.values, encode_labels(sources_train)
X_valid, Y_valid = titles_valid.values, encode_labels(sources_valid)
```

In [28]:
```
X_train[:3]
```

Out[28]:
```
array(['clj3d  a clojure graphic library ', 'factory girl for scala',
       'searching for meaningful markers of aging'], dtype=object)
```

In [29]:
```
Y_train[:3]
```

Out[29]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)
```

NNLM Model

We will first try a word embedding pre-trained using a Neural Probabilistic Language Model. TF-Hub has a 50-dimensional one called nnlm-en-dim50-with-normalization, which also normalizes the vectors produced.

Once loaded from its url, the TF-hub module can be used as a normal Keras layer in a sequential or functional model. Since we have enough data to fine-tune the parameters of the pre-trained embedding itself, we will set `trainable=True` in the `KerasLayer` that loads the pre-trained embedding:

In [32]:
```
NNLM = "https://tfhub.dev/google/nnlm-en-dim50/2"

nnlm_module = KerasLayer(
    NNLM, output_shape=[50], input_shape=[], dtype=tf.string, trainable=True)
```

Note that this TF-Hub embedding produces a single 50-dimensional vector when passed a

sentence:

In [33]:
```
nnlm_module(tf.constant(["The dog is happy to see people in the street."]))
```

Out[33]:
```
<tf.Tensor: shape=(1, 50), dtype=float32, numpy=
array([[ 0.19331802,  0.05893906,  0.15330684,  0.2505918 ,  0.19369544,
         0.03578748,  0.07387847, -0.10962156, -0.11377034,  0.07172022,
         0.12458669, -0.02289705, -0.18177685, -0.07084437, -0.00225849,
        -0.36875236,  0.05772953, -0.14222091,  0.08765972, -0.14068899,
        -0.07005888, -0.20634466,  0.07220475,  0.04258814,  0.0955702 ,
         0.19424029, -0.42492998, -0.00706906, -0.02095   , -0.05055764,
        -0.18988201, -0.02841404,  0.13222624, -0.01459922, -0.31255388,
        -0.09577855,  0.05469003, -0.13858607,  0.01141668, -0.12352604,
        -0.07250367, -0.11605677, -0.06976165,  0.14313601, -0.15183711,
        -0.06836402,  0.03054246, -0.13259597, -0.14599673,  0.05094011]],
      dtype=float32)>
```

Building the models

Let's write a function that

- takes as input an instance of a `KerasLayer` (i.e. the `nnlm_module` we constructed above) as well as the name of the model (say `nnlm`)
- returns a compiled Keras sequential model starting with this pre-trained TF-hub layer, adding one or more dense relu layers to it, and ending with a softmax layer giving the probability of each of the classes:

In [35]:
```
def build_model(hub_module, name):
    model = Sequential([
        hub_module,
        Dense(16, activation='relu'),
        Dense(N_CLASSES, activation='softmax')
    ], name=name)

    model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Let's also wrap the training code into a `train_and_evaluate` function that

- takes as input the training and validation data, as well as the compiled model itself, and the `batch_size`
- trains the compiled model for 100 epochs at most, and does early-stopping when the validation loss is no longer decreasing
- returns an `history` object, which will help us to plot the learning curves

In [48]:
```
def train_and_evaluate(train_data, val_data, model, batch_size=5000):
    X_train, Y_train = train_data

    tf.random.set_seed(33)
```

```
    model_dir = os.path.join(MODEL_DIR, model.name)
    if tf.io.gfile.exists(model_dir):
        tf.io.gfile.rmtree(model_dir)

    history = model.fit(
        X_train, Y_train,
        epochs=200,
        batch_size=batch_size,
        validation_data=val_data,
        callbacks=[EarlyStopping(), TensorBoard(model_dir)],
    )
    return history
```

Training NNLM

In [49]:
```
data = (X_train, Y_train)
val_data = (X_valid, Y_valid)
```

In [50]:
```
nnlm_model = build_model(nnlm_module, 'nnlm')
nnlm_history = train_and_evaluate(data, val_data, nnlm_model)
```

```
2021-10-29 13:48:44.665800: I tensorflow/core/profiler/lib/profiler_session.cc:1
31] Profiler session initializing.
2021-10-29 13:48:44.665836: I tensorflow/core/profiler/lib/profiler_session.cc:1
46] Profiler session started.
2021-10-29 13:48:44.665995: I tensorflow/core/profiler/lib/profiler_session.cc:1
64] Profiler session tear down.
Epoch 1/200
 1/19 [>.............................] - ETA: 15s - loss: 0.9710 - accuracy: 0.7
242
2021-10-29 13:48:45.649792: I tensorflow/core/profiler/lib/profiler_session.cc:1
31] Profiler session initializing.
2021-10-29 13:48:45.649842: I tensorflow/core/profiler/lib/profiler_session.cc:1
46] Profiler session started.
 2/19 [==>...........................] - ETA: 7s - loss: 0.9687 - accuracy: 0.72
78
2021-10-29 13:48:46.083562: I tensorflow/core/profiler/lib/profiler_session.cc:6
6] Profiler session collecting data.
2021-10-29 13:48:46.085718: I tensorflow/core/profiler/lib/profiler_session.cc:1
64] Profiler session tear down.
2021-10-29 13:48:46.087762: I tensorflow/core/profiler/rpc/client/save_profile.c
c:136] Creating directory: ./text_models/nnlm/train/plugins/profile/2021_10_29_1
3_48_46

2021-10-29 13:48:46.088916: I tensorflow/core/profiler/rpc/client/save_profile.c
c:142] Dumped gzipped tool data for trace.json.gz to ./text_models/nnlm/train/pl
ugins/profile/2021_10_29_13_48_46/tensorflow-2-6-20211029-092628.trace.json.gz
2021-10-29 13:48:46.093415: I tensorflow/core/profiler/rpc/client/save_profile.c
c:136] Creating directory: ./text_models/nnlm/train/plugins/profile/2021_10_29_1
3_48_46

2021-10-29 13:48:46.094336: I tensorflow/core/profiler/rpc/client/save_profile.c
c:142] Dumped gzipped tool data for memory_profile.json.gz to ./text_models/nnl
m/train/plugins/profile/2021_10_29_13_48_46/tensorflow-2-6-20211029-092628.memor
y_profile.json.gz
2021-10-29 13:48:46.094763: I tensorflow/core/profiler/rpc/client/capture_profil
```

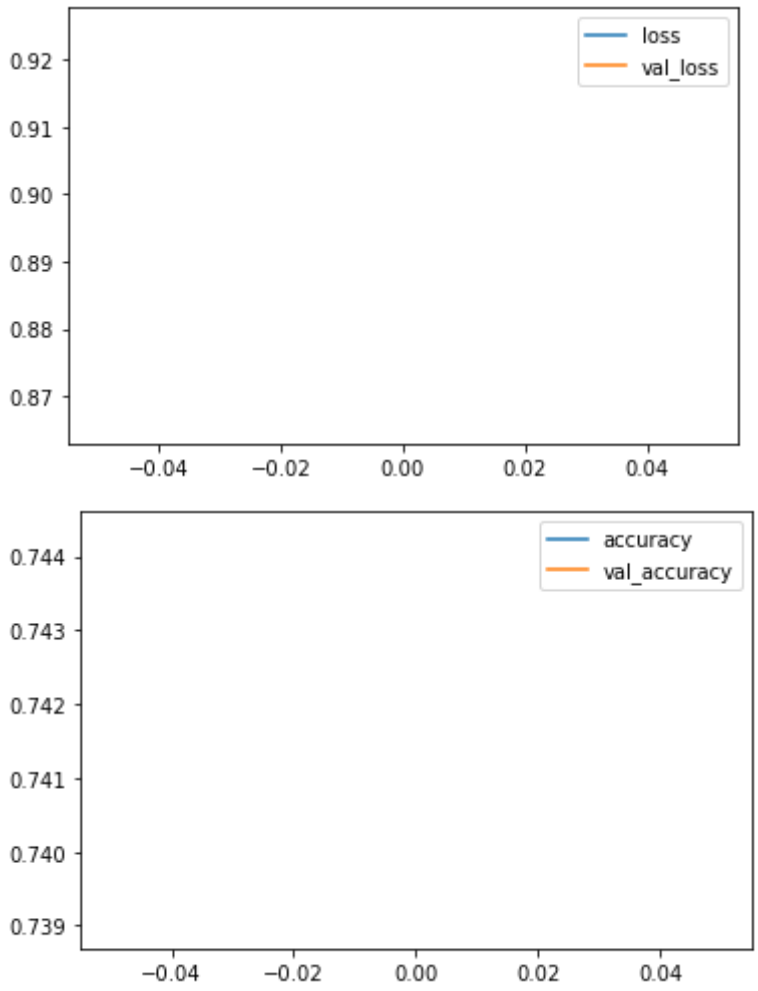
```
e.cc:251] Creating directory: ./text_models/nnlm/train/plugins/profile/2021_10_2
9_13_48_46
Dumped tool data for xplane.pb to ./text_models/nnlm/train/plugins/profile/2021_
10_29_13_48_46/tensorflow-2-6-20211029-092628.xplane.pb
Dumped tool data for overview_page.pb to ./text_models/nnlm/train/plugins/profil
e/2021_10_29_13_48_46/tensorflow-2-6-20211029-092628.overview_page.pb
Dumped tool data for input_pipeline.pb to ./text_models/nnlm/train/plugins/profi
le/2021_10_29_13_48_46/tensorflow-2-6-20211029-092628.input_pipeline.pb
Dumped tool data for tensorflow_stats.pb to ./text_models/nnlm/train/plugins/pro
file/2021_10_29_13_48_46/tensorflow-2-6-20211029-092628.tensorflow_stats.pb
Dumped tool data for kernel_stats.pb to ./text_models/nnlm/train/plugins/profil
e/2021_10_29_13_48_46/tensorflow-2-6-20211029-092628.kernel_stats.pb

19/19 [==============================] - 8s 370ms/step - loss: 0.9248 - accurac
y: 0.7390 - val_loss: 0.8659 - val_accuracy: 0.7443
```

In [51]:

```
history = nnlm_history
pd.DataFrame(history.history)[['loss', 'val_loss']].plot()
pd.DataFrame(history.history)[['accuracy', 'val_accuracy']].plot()
```

Out[51]:

```
<AxesSubplot:>
```



Bonus

Try to beat the best model by modifying the model architecture, changing the TF-Hub embedding, and tweaking the training parameters.