In [1]:

```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Copyright 2019 The TensorFlow Authors.

In [2]:

```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

In [3]:

```
import csv
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

!wget --no-check-certificate \
    https://storage.googleapis.com/laurencemoroney-blog.appspot.com/bbc-text.csv \
    -O /tmp/bbc-text.csv
```

```
--2020-10-06 12:34:16--  https://storage.googleapis.com/laurencemoroney-blog.appspot.com/bbc-text.csv
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.203.128, 74.125.204.128, 64.233.188.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.203.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5057493 (4.8M) [application/octet-stream]
Saving to: ‘/tmp/bbc-text.csv’

/tmp/bbc-text.csv   100%[===================>]   4.82M  --.-KB/s    in 0.04s

2020-10-06 12:34:16 (138 MB/s) - ‘/tmp/bbc-text.csv’ saved [5057493/5057493]
```

In [4]:

```
vocab_size = 1000
embedding_dim = 16
max_length = 120
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"
training_portion = .8
```

In [5]:

```
sentences = []
labels = []
stopwords = [ "a", "about", "above", "after", "again", "against", "all", "am", "an", "and", "any",
"are", "as", "at", "be", "because", "been", "before", "being", "below", "between", "both", "but", "
by", "could", "did", "do", "does", "doing", "down", "during", "each", "few", "for", "from",
"further", "had", "has", "have", "having", "he", "he'd", "he'll", "he's", "her", "here", "here's",
"hers", "herself", "him", "himself", "his", "how", "how's", "i", "i'd", "i'll", "i'm", "i've", "if"
, "in", "into", "is", "it", "it's", "its", "itself", "let's", "me", "more", "most", "my", "myself",
"nor", "of", "on", "once", "only", "or", "other", "ought", "our", "ours", "ourselves", "out", "over
", "own", "same", "she", "she'd", "she'll", "she's", "should", "so", "some", "such", "than", "that"
, "that's", "the", "their", "theirs", "them", "themselves", "then", "there", "there's", "these", "t
hey", "they'd", "they'll", "they're", "they've", "this", "those", "through", "to", "too", "under",
"until", "up", "very", "was", "we", "we'd", "we'll", "we're", "we've", "were", "what", "what's", "w
hen", "when's", "where", "where's", "which", "while", "who", "who's", "whom", "why", "why's", "with
", "would", "you", "you'd", "you'll", "you're", "you've", "your", "yours", "yourself", "yourselves"
]
print(len(stopwords))
# Expected Output
# 153
```

153

In [6]:

```
with open("/tmp/bbc-text.csv", 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    next(reader)
    for row in reader:
        labels.append(row[0])
        sentence = row[1]
        for word in stopwords:
            token = " " + word + " "
            sentence = sentence.replace(token, " ")
        sentences.append(sentence)

print(len(labels))
print(len(sentences))
print(sentences[0])
# Expected Output
# 2225
# 2225
# tv future hands viewers home theatre systems  plasma high-definition tvs  digital video
recorders moving living room  way people watch tv will radically different five years  time.  acco
rding expert panel gathered annual consumer electronics show las vegas discuss new technologies wi
ll impact one favourite pastimes. us leading trend  programmes content will delivered viewers via
home networks  cable  satellite  telecoms companies  broadband service providers front rooms porta
ble devices.  one talked-about technologies ces digital personal video recorders (dvr pvr). set-to
p boxes  like us s tivo uk s sky+ system  allow people record  store  play  pause forward wind tv
programmes want.  essentially  technology allows much personalised tv. also built-in high-definiti
on tv sets  big business japan us  slower take off europe lack high-definition programming. not ca
n people forward wind adverts  can also forget abiding network channel schedules  putting together
a-la-carte entertainment. us networks cable satellite companies worried means terms advertising re
venues well  brand identity  viewer loyalty channels. although us leads technology moment  also co
ncern raised europe  particularly growing uptake services like sky+.  happens today  will see nine
months years  time uk  adam hume  bbc broadcast s futurologist told bbc news website. likes bbc
no issues lost advertising revenue yet. pressing issue moment commercial uk broadcasters  brand lo
yalty important everyone.  will talking content brands rather network brands  said tim hanlon  br
and communications firm starcom mediavest.  reality broadband connections  anybody can producer co
ntent.  added:  challenge now hard promote programme much choice.  means  said stacey jolna
senior vice president tv guide tv group  way people find content want watch simplified tv viewers.
means networks  us terms  channels take leaf google s book search engine future  instead scheduler
help people find want watch. kind channel model might work younger ipod generation used taking con
trol gadgets play them. might not suit everyone  panel recognised. older generations comfortable f
amiliar schedules channel brands know getting. perhaps not want much choice put hands  mr hanlon s
uggested.  end  kids just diapers pushing buttons already - everything possible available  said m
r hanlon.  ultimately  consumer will tell market want.  50 000 new gadgets technologies showcased
ces  many enhancing tv-watching experience. high-definition tv sets everywhere many new models lcd
(liquid crystal display) tvs launched dvr capability built  instead external boxes. one example la
unched show humax s 26-inch lcd tv 80-hour tivo dvr dvd recorder. one us s biggest satellite tv co
mpanies  directtv  even launched branded dvr show 100-hours recording capability  instant replay
search function. set can pause rewind tv 90 hours. microsoft chief bill gates announced pre-show k
eynote speech partnership tivo  called tivotogo  means people can play recorded programmes windows
```

```
pcs mobile devices. reflect increasing trend freeing multimedia people can watch want  want.
```

```
2225
2225
tv future hands viewers home theatre systems  plasma high-definition tvs  digital video recorders
moving living room  way people watch tv will radically different five years  time.  according
expert panel gathered annual consumer electronics show las vegas discuss new technologies will imp
act one favourite pastimes. us leading trend  programmes content will delivered viewers via home n
etworks  cable  satellite  telecoms companies  broadband service providers front rooms portable
devices.  one talked-about technologies ces digital personal video recorders (dvr pvr). set-top bo
xes  like us s tivo uk s sky+ system  allow people record  store  play  pause forward wind tv progr
ammes want.  essentially  technology allows much personalised tv. also built-in high-definition tv
sets  big business japan us  slower take off europe lack high-definition programming. not can peop
le forward wind adverts  can also forget abiding network channel schedules  putting together a-la-
carte entertainment. us networks cable satellite companies worried means terms advertising
revenues well  brand identity  viewer loyalty channels. although us leads technology moment  also
concern raised europe  particularly growing uptake services like sky+.  happens today  will see ni
ne months years  time uk  adam hume  bbc broadcast s futurologist told bbc news website. likes bb
c  no issues lost advertising revenue yet. pressing issue moment commercial uk broadcasters  brand
loyalty important everyone.  will talking content brands rather network brands  said tim hanlon
brand communications firm starcom mediavest.  reality broadband connections  anybody can producer
content.  added:  challenge now hard promote programme much choice.  means  said stacey jolna
senior vice president tv guide tv group  way people find content want watch simplified tv viewers.
means networks  us terms  channels take leaf google s book search engine future  instead scheduler
help people find want watch. kind channel model might work younger ipod generation used taking con
trol gadgets play them. might not suit everyone  panel recognised. older generations comfortable f
amiliar schedules channel brands know getting. perhaps not want much choice put hands  mr hanlon s
uggested.  end  kids just diapers pushing buttons already - everything possible available  said m
r hanlon.  ultimately  consumer will tell market want.  50 000 new gadgets technologies showcased
ces  many enhancing tv-watching experience. high-definition tv sets everywhere many new models lcd
(liquid crystal display) tvs launched dvr capability built  instead external boxes. one example la
unched show humax s 26-inch lcd tv 80-hour tivo dvr dvd recorder. one us s biggest satellite tv co
mpanies  directtv  even launched branded dvr show 100-hours recording capability  instant replay
search function. set can pause rewind tv 90 hours. microsoft chief bill gates announced pre-show k
eynote speech partnership tivo  called tivotogo  means people can play recorded programmes windows
pcs mobile devices. reflect increasing trend freeing multimedia people can watch want  want.
```

In [7]:

```
train_size = int(len(sentences) * training_portion)

train_sentences = sentences[:train_size]
train_labels = labels[:train_size]

validation_sentences = sentences[train_size:]
validation_labels = labels[train_size:]

print(train_size)
print(len(train_sentences))
print(len(train_labels))
print(len(validation_sentences))
print(len(validation_labels))

# Expected output (if training_portion=.8)
# 1780
# 1780
# 1780
# 445
# 445
```

```
1780
1780
1780
445
445
```

In [8]:

```
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_sentences)
word_index = tokenizer.word_index

train_sequences = tokenizer.texts_to_sequences(train_sentences)
train_padded = pad_sequences(train_sequences, padding=padding_type, maxlen=max_length)
```

```
train_padded = pad_sequences(train_sequences, padding=padding_type, maxlen=max_length)

print(len(train_sequences[0]))
print(len(train_padded[0]))

print(len(train_sequences[1]))
print(len(train_padded[1]))

print(len(train_sequences[10]))
print(len(train_padded[10]))

# Expected Ouput
# 449
# 120
# 200
# 120
# 192
# 120
```

```
449
120
200
120
192
120
```

In [9]:

```
validation_sequences = tokenizer.texts_to_sequences(validation_sentences)
validation_padded = pad_sequences(validation_sequences, padding=padding_type, maxlen=max_length)

print(len(validation_sequences))
print(validation_padded.shape)

# Expected output
# 445
# (445, 120)
```

```
445
(445, 120)
```

In [10]:

```
label_tokenizer = Tokenizer()
label_tokenizer.fit_on_texts(labels)

training_label_seq = np.array(label_tokenizer.texts_to_sequences(train_labels))
validation_label_seq = np.array(label_tokenizer.texts_to_sequences(validation_labels))

print(training_label_seq[0])
print(training_label_seq[1])
print(training_label_seq[2])
print(training_label_seq.shape)

print(validation_label_seq[0])
print(validation_label_seq[1])
print(validation_label_seq[2])
print(validation_label_seq.shape)

# Expected output
# [4]
# [2]
# [1]
# (1780, 1)
# [5]
# [4]
# [3]
# (445, 1)
```

```
[4]
[2]
[1]
(1780, 1)
[5]
```

```
[5]
[4]
[3]
(445, 1)
```

In [11]:

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(6, activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()

# Expected Output
# Layer (type)                 Output Shape              Param #
# =================================================================
# embedding (Embedding)        (None, 120, 16)           16000
# _________________________________________________________________
# global_average_pooling1d (Gl (None, 16)                0
# _________________________________________________________________
# dense (Dense)                (None, 24)                408
# _________________________________________________________________
# dense_1 (Dense)              (None, 6)                 150
# =================================================================
# Total params: 16,558
# Trainable params: 16,558
# Non-trainable params: 0
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 120, 16)           16000
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 24)                408
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 150
=================================================================
Total params: 16,558
Trainable params: 16,558
Non-trainable params: 0
_________________________________________________________________
```

In [12]:

```
num_epochs = 30
history = model.fit(train_padded, training_label_seq, epochs=num_epochs, validation_data=(validatio
n_padded, validation_label_seq), verbose=2)
```

```
Epoch 1/30
56/56 - 0s - loss: 1.7692 - accuracy: 0.3202 - val_loss: 1.7358 - val_accuracy: 0.2382
Epoch 2/30
56/56 - 0s - loss: 1.6973 - accuracy: 0.2281 - val_loss: 1.6463 - val_accuracy: 0.2584
Epoch 3/30
56/56 - 0s - loss: 1.5971 - accuracy: 0.3764 - val_loss: 1.5345 - val_accuracy: 0.4472
Epoch 4/30
56/56 - 0s - loss: 1.4621 - accuracy: 0.5275 - val_loss: 1.3865 - val_accuracy: 0.5910
Epoch 5/30
56/56 - 0s - loss: 1.2868 - accuracy: 0.5994 - val_loss: 1.2040 - val_accuracy: 0.6449
Epoch 6/30
56/56 - 0s - loss: 1.0873 - accuracy: 0.6607 - val_loss: 1.0145 - val_accuracy: 0.7596
Epoch 7/30
56/56 - 0s - loss: 0.8927 - accuracy: 0.8326 - val_loss: 0.8450 - val_accuracy: 0.8404
Epoch 8/30
56/56 - 0s - loss: 0.7274 - accuracy: 0.8871 - val_loss: 0.7086 - val_accuracy: 0.8876
Epoch 9/30
56/56 - 0s - loss: 0.5889 - accuracy: 0.9287 - val_loss: 0.5892 - val_accuracy: 0.8989
Epoch 10/30
56/56 - 0s - loss: 0.4717 - accuracy: 0.9365 - val_loss: 0.4998 - val_accuracy: 0.9124
```

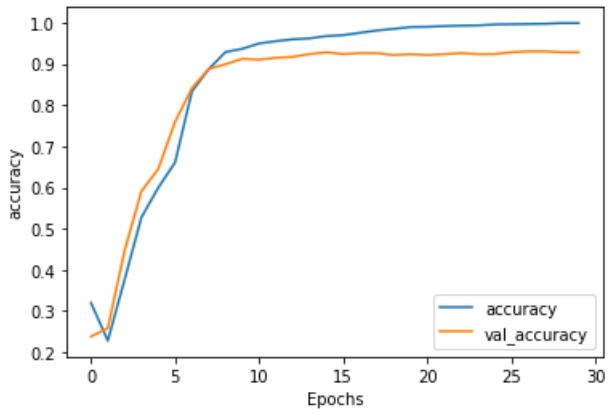
```
Epoch 11/30
56/56 - 0s - loss: 0.3747 - accuracy: 0.9494 - val_loss: 0.4125 - val_accuracy: 0.9101
Epoch 12/30
56/56 - 0s - loss: 0.3004 - accuracy: 0.9551 - val_loss: 0.3570 - val_accuracy: 0.9146
Epoch 13/30
56/56 - 0s - loss: 0.2470 - accuracy: 0.9596 - val_loss: 0.3149 - val_accuracy: 0.9169
Epoch 14/30
56/56 - 0s - loss: 0.2096 - accuracy: 0.9618 - val_loss: 0.2893 - val_accuracy: 0.9236
Epoch 15/30
56/56 - 0s - loss: 0.1800 - accuracy: 0.9674 - val_loss: 0.2661 - val_accuracy: 0.9281
Epoch 16/30
56/56 - 0s - loss: 0.1573 - accuracy: 0.9697 - val_loss: 0.2508 - val_accuracy: 0.9236
Epoch 17/30
56/56 - 0s - loss: 0.1388 - accuracy: 0.9753 - val_loss: 0.2368 - val_accuracy: 0.9258
Epoch 18/30
56/56 - 0s - loss: 0.1234 - accuracy: 0.9809 - val_loss: 0.2289 - val_accuracy: 0.9258
Epoch 19/30
56/56 - 0s - loss: 0.1097 - accuracy: 0.9848 - val_loss: 0.2205 - val_accuracy: 0.9213
Epoch 20/30
56/56 - 0s - loss: 0.0980 - accuracy: 0.9893 - val_loss: 0.2128 - val_accuracy: 0.9236
Epoch 21/30
56/56 - 0s - loss: 0.0877 - accuracy: 0.9899 - val_loss: 0.2063 - val_accuracy: 0.9213
Epoch 22/30
56/56 - 0s - loss: 0.0792 - accuracy: 0.9916 - val_loss: 0.2028 - val_accuracy: 0.9236
Epoch 23/30
56/56 - 0s - loss: 0.0712 - accuracy: 0.9927 - val_loss: 0.1999 - val_accuracy: 0.9258
Epoch 24/30
56/56 - 0s - loss: 0.0644 - accuracy: 0.9933 - val_loss: 0.1944 - val_accuracy: 0.9236
Epoch 25/30
56/56 - 0s - loss: 0.0579 - accuracy: 0.9955 - val_loss: 0.1921 - val_accuracy: 0.9236
Epoch 26/30
56/56 - 0s - loss: 0.0525 - accuracy: 0.9961 - val_loss: 0.1902 - val_accuracy: 0.9281
Epoch 27/30
56/56 - 0s - loss: 0.0474 - accuracy: 0.9966 - val_loss: 0.1872 - val_accuracy: 0.9303
Epoch 28/30
56/56 - 0s - loss: 0.0435 - accuracy: 0.9972 - val_loss: 0.1879 - val_accuracy: 0.9303
Epoch 29/30
56/56 - 0s - loss: 0.0391 - accuracy: 0.9989 - val_loss: 0.1845 - val_accuracy: 0.9281
Epoch 30/30
56/56 - 0s - loss: 0.0355 - accuracy: 0.9989 - val_loss: 0.1851 - val_accuracy: 0.9281
```

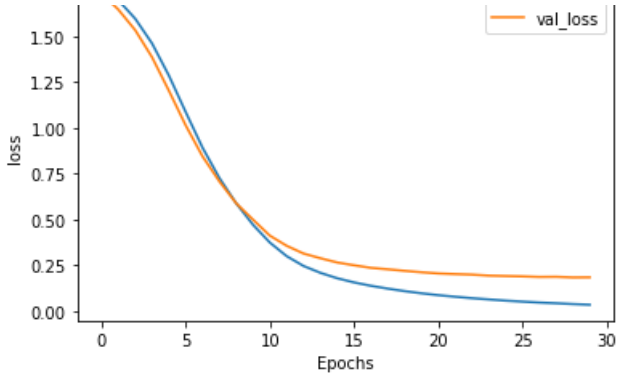
In [13]:

```
import matplotlib.pyplot as plt


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

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```





In [14]:

```
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])

def decode_sentence(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])
```

In [15]:

```
e = model.layers[0]
weights = e.get_weights()[0]
print(weights.shape) # shape: (vocab_size, embedding_dim)

# Expected output
# (1000, 16)
```

```
(1000, 16)
```

In [16]:

```
import io

out_v = io.open('vecs.tsv', 'w', encoding='utf-8')
out_m = io.open('meta.tsv', 'w', encoding='utf-8')
for word_num in range(1, vocab_size):
  word = reverse_word_index[word_num]
  embeddings = weights[word_num]
  out_m.write(word + "\n")
  out_v.write('\t'.join([str(x) for x in embeddings]) + "\n")
out_v.close()
out_m.close()
```

In [17]:

```
try:
  from google.colab import files
except ImportError:
  pass
else:
  files.download('vecs.tsv')
  files.download('meta.tsv')
```

In [ ]:

```
```