In [1]:
```
pip install spacy
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.6/d
ist-packages (2.2.4)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/l
ib/python3.6/dist-packages (from spacy) (2.0.4)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/loc
al/lib/python3.6/dist-packages (from spacy) (2.23.0)
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/pyth
on3.6/dist-packages (from spacy) (7.4.0)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/li
b/python3.6/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/l
ib/python3.6/dist-packages (from spacy) (4.41.1)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /usr/local/l
ib/python3.6/dist-packages (from spacy) (1.0.2)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/l
ocal/lib/python3.6/dist-packages (from spacy) (1.0.3)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/loc
al/lib/python3.6/dist-packages (from spacy) (1.0.0)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/pyt
hon3.6/dist-packages (from spacy) (1.18.5)
Requirement already satisfied: setuptools in /usr/local/lib/python
3.6/dist-packages (from spacy) (50.3.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local
/lib/python3.6/dist-packages (from spacy) (3.0.2)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in /usr/local/li
b/python3.6/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/
lib/python3.6/dist-packages (from spacy) (0.8.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/li
b/python3.6/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2
020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.
21.1 in /usr/local/lib/python3.6/dist-packages (from requests<3.0.
0,>=2.13.0->spacy) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib
/python3.6/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.
0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/pyth
on3.6/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.10)
Requirement already satisfied: importlib-metadata>=0.20; python_ve
rsion < "3.8" in /usr/local/lib/python3.6/dist-packages (from cata
logue<1.1.0,>=0.0.7->spacy) (2.0.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3
.6/dist-packages (from importlib-metadata>=0.20; python_version <
"3.8"->catalogue<1.1.0,>=0.0.7->spacy) (3.4.0)
```

Tokenizing

In [5]:
```
import spacy
```

In [8]:
```
nlp = spacy.load("en_core_web_sm")
```

In [9]:
```
text = """
Dave watched as the forest burned up on the hill,
only a few miles from his house. The car had
been hastily packed and Marta was inside trying to round
up the last of the pets. "Where could she be?" he wondered
as he continued to wait for Marta to appear with the pets.
"""
```

In [10]:
```
nlp = spacy.load("en_core_web_sm")
```

In [11]:
```
doc = nlp(text)
```

In [12]:
```
token_list = [token for token in doc]
```

In [18]:
```
print(token_list)
print(len(token_list))
```

```
[
, Dave, watched, as, the, forest, burned, up, on, the, hill, ,,
, only, a, few, miles, from, his, house, ., The, car, had,
, been, hastily, packed, and, Marta, was, inside, trying, to, roun
d,
, up, the, last, of, the, pets, ., ", Where, could, she, be, ?, ",
he, wondered,
, as, he, continued, to, wait, for, Marta, to, appear, with, the,
pets, .,
]
67
```

Removing Stop Words

In [15]:
```
filtered_tokens = [token for token in doc if not token.is_stop]
```

In [19]:
```
print(filtered_tokens)
print(len(filtered_tokens))
```

```
[
, Dave, watched, forest, burned, hill, ,,
, miles, house, ., car,
, hastily, packed, Marta, inside, trying, round,
, pets, ., ", ?, ", wondered,
, continued, wait, Marta, appear, pets, .,
]
34
```

Normalizing Words

In [20]:
```
lemmas = [
    f"Token: {token}, lemma: {token.lemma_}"
    for token in filtered_tokens
]
```

In [21]:
```
print(lemmas)
```

```
['Token: \n, lemma: \n', 'Token: Dave, lemma: Dave', 'Token: watch
ed, lemma: watch', 'Token: forest, lemma: forest', 'Token: burned,
lemma: burn', 'Token: hill, lemma: hill', 'Token: ,, lemma: ,', 'T
oken: \n, lemma: \n', 'Token: miles, lemma: mile', 'Token: house,
lemma: house', 'Token: ., lemma: .', 'Token: car, lemma: car', 'To
ken: \n, lemma: \n', 'Token: hastily, lemma: hastily', 'Token: pac
ked, lemma: pack', 'Token: Marta, lemma: Marta', 'Token: inside, l
emma: inside', 'Token: trying, lemma: try', 'Token: round, lemma:
round', 'Token: \n, lemma: \n', 'Token: pets, lemma: pet', 'Token:
., lemma: .', 'Token: ", lemma: "', 'Token: ?, lemma: ?', 'Token:
", lemma: "', 'Token: wondered, lemma: wonder', 'Token: \n, lemma:
\n', 'Token: continued, lemma: continue', 'Token: wait, lemma: wai
t', 'Token: Marta, lemma: Marta', 'Token: appear, lemma: appear',
'Token: pets, lemma: pet', 'Token: ., lemma: .', 'Token: \n, lemma
: \n']
```

In [55]:
```
filtered_tokens[1].vector
```

```
Out[55]: array([ 1.6193167e+00, -2.7117019e+00, -6.8552375e-01,  2.6652899e
+00,
         4.5226312e+00,  2.8338575e+00,  6.1740106e-01,  9.5401168e
-01,
         2.6201737e+00,  2.5994289e+00,  5.9061027e+00, -1.7552420e
-01,
        -8.7880111e-01,  4.8553795e-03, -1.7236035e+00, -1.7494547e
+00,
        -1.0313329e+00,  1.6518956e-01,  5.3024960e-01, -3.2018152e
-01,
        -2.6411371e+00, -2.4750671e+00, -5.0014794e-01, -3.3213449e
+00,
        -5.3300351e-01,  2.3968523e+00,  1.5485952e+00, -2.2231889e
+00,
        -1.2597762e+00, -5.6858027e-01, -9.4768405e-02, -1.3759263e
+00,
        -1.0165324e+00,  5.6860483e-01,  2.6817162e+00, -3.7418640e
+00,
         2.7644300e+00, -1.9967061e+00, -2.9627855e+00, -1.0863459e
-01,
         2.7437925e+00,  2.5450244e+00,  1.6124392e+00, -3.3037057e
+00,
        -2.4419413e+00,  9.5868981e-01,  1.1957375e+00, -1.2429583e
+00,
        -1.2961357e+00,  2.8916957e+00, -2.8091950e+00, -3.1826324e
+00,
        -2.4809690e+00, -2.5254309e-01, -2.0454383e+00,  3.0948038e
+00,
         2.3146892e+00,  3.1973858e+00, -1.0000327e+00, -1.8173008e
+00,
        -1.8257004e-01, -1.3169163e-01, -1.6473753e+00, -3.0071383e
+00,
         5.8041401e+00, -2.8103060e-01,  1.7717384e-01, -2.2952008e
+00,
        -1.3665587e+00,  4.3192568e-01,  1.5252322e+00,  1.0701846e
+00,
         4.5825213e-01, -1.7640622e+00,  1.0941651e+00, -3.9024787e
+00,
         2.3232937e-02, -4.5654988e-01, -2.3950067e+00,  1.0093416e
+00,
         3.8597989e+00, -1.0375429e+00, -2.4387794e+00, -1.1583717e
+00,
         3.8506849e+00,  3.5678258e+00, -2.9660366e+00, -3.1863713e
+00,
         4.5841753e-02,  8.6235547e-01, -1.8335643e+00, -1.6974587e
+00,
        -1.1221293e+00,  2.0094342e+00,  3.8732340e+00,  2.9344873e
+00],
      dtype=float32)
```

Making The Model

In [23]:
```
import tensorflow as tf
from tensorflow import keras

import numpy as np

print(tf.__version__)
```

```
2.3.0
```

In [56]:
```
import matplotlib.pyplot as plt
import os
import re
import shutil
import string
import tensorflow as tf

from tensorflow.keras import layers
from tensorflow.keras import losses
from tensorflow.keras import preprocessing
from tensorflow.keras.layers.experimental.preprocessing import Text
Vectorization
```

In [57]:
```
url = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar
.gz"

dataset = tf.keras.utils.get_file("aclImdb_v1.tar.gz", url,
                                  untar=True, cache_dir='.',
                                  cache_subdir='')

dataset_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')
```

```
Downloading data from https://ai.stanford.edu/~amaas/data/sentimen
t/aclImdb_v1.tar.gz
84131840/84125825 [==============================] - 5s 0us/step
```

In [58]:
```
os.listdir(dataset_dir)
```

Out[58]:
```
['test', 'train', 'README', 'imdbEr.txt', 'imdb.vocab']
```

In [59]:
```
train_dir = os.path.join(dataset_dir, 'train')
os.listdir(train_dir)
```

Out[59]:
```
['urls_neg.txt',
 'urls_unsup.txt',
 'unsup',
 'neg',
 'urls_pos.txt',
 'labeledBow.feat',
 'unsupBow.feat',
 'pos']
```

In [60]:
```
sample_file = os.path.join(train_dir, 'pos/1181_9.txt')
with open(sample_file) as f:
  print(f.read())
```

```
Rachel Griffiths writes and directs this award winning short film.
A heartwarming story about coping with grief and cherishing the me
mory of those we've loved and lost. Although, only 15 minutes long
, Griffiths manages to capture so much emotion and truth onto film
in the short space of time. Bud Tingwell gives a touching performa
nce as Will, a widower struggling to cope with his wife's death. W
ill is confronted by the harsh reality of loneliness and helplessn
ess as he proceeds to take care of Ruth's pet cow, Tulip. The film
displays the grief and responsibility one feels for those they hav
e loved and lost. Good cinematography, great direction, and superb
ly acted. It will bring tears to all those who have lost a loved o
ne, and survived.
```

In [62]:
```
remove_dir = os.path.join(train_dir, 'unsup')
shutil.rmtree(remove_dir)
```

In [63]:
```
batch_size = 32
seed = 42

raw_train_ds = tf.keras.preprocessing.text_dataset_from_directory(
    'aclImdb/train',
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=seed)
```

```
Found 25000 files belonging to 2 classes.
Using 20000 files for training.
```

In [64]:
```
for text_batch, label_batch in raw_train_ds.take(1):
  for i in range(3):
    print("Review", text_batch.numpy()[i])
    print("Label", label_batch.numpy()[i])
```

```
Review b'"Pandemonium" is a horror movie spoof that comes off more
stupid than funny. Believe me when I tell you, I love comedies. Es
pecially comedy spoofs. "Airplane", "The Naked Gun" trilogy, "Blaz
ing Saddles", "High Anxiety", and "Spaceballs" are some of my favo
rite comedies that spoof a particular genre. "Pandemonium" is not
up there with those films. Most of the scenes in this movie had me
sitting there in stunned silence because the movie wasn\'t all tha
t funny. There are a few laughs in the film, but when you watch a
comedy, you expect to laugh a lot more than a few times and that\'
s all this film has going for it. Geez, "Scream" had more laughs t
han this film and that was more of a horror film. How bizarre is t
hat?<br /><br />*1/2 (out of four)'
Label 0
Review b"David Mamet is a very interesting and a very un-equal dir
ector. His first movie 'House of Games' was the one I liked best,
and it set a series of films with characters whose perspective of
life changes as they get into complicated situations, and so does
the perspective of the viewer.<br /><br />So is 'Homicide' which f
rom the title tries to set the mind of the viewer to the usual cri
me drama. The principal characters are two cops, one Jewish and on
e Irish who deal with a racially charged area. The murder of an ol
d Jewish shop owner who proves to be an ancient veteran of the Isr
aeli Independence war triggers the Jewish identity in the mind and
heart of the Jewish detective.<br /><br />This is were the flaws o
f the film are the more obvious. The process of awakening is theat
rical and hard to believe, the group of Jewish militants is operat
ic, and the way the detective eventually walks to the final violen
t confrontation is pathetic. The end of the film itself is Mamet-l
ike smart, but disappoints from a human emotional perspective.<br
/><br />Joe Mantegna and William Macy give strong performances, bu
t the flaws of the story are too evident to be easily compensated.
"
Label 0
Review b'Great documentary about the lives of NY firefighters duri
ng the worst terrorist attack of all time.. That reason alone is w
hy this should be a must see collectors item.. What shocked me was
not only the attacks, but the"High Fat Diet" and physical appearan
ce of some of these firefighters. I think a lot of Doctors would a
gree with me that,in the physical shape they were in, some of thes
e firefighters would NOT of made it to the 79th floor carrying ove
r 60 lbs of gear. Having said that i now have a greater respect fo
r firefighters and i realize becoming a firefighter is a life alte
ring job. The French have a history of making great documentary\'s
and that is what this is, a Great Documentary.....'
Label 1
```

In [65]:
```
print("Label 0 corresponds to", raw_train_ds.class_names[0])
print("Label 1 corresponds to", raw_train_ds.class_names[1])
```

```
Label 0 corresponds to neg
Label 1 corresponds to pos
```

In [66]:
```
raw_val_ds = tf.keras.preprocessing.text_dataset_from_directory(
    'aclImdb/train',
    batch_size=batch_size,
    validation_split=0.2,
    subset='validation',
    seed=seed)
```

```
Found 25000 files belonging to 2 classes.
Using 5000 files for validation.
```

In [67]:
```
raw_test_ds = tf.keras.preprocessing.text_dataset_from_directory(
    'aclImdb/test',
    batch_size=batch_size)
```

```
Found 25000 files belonging to 2 classes.
```

In [68]:
```
def custom_standardization(input_data):
  lowercase = tf.strings.lower(input_data)
  stripped_html = tf.strings.regex_replace(lowercase, '<br />', ' '
)
  return tf.strings.regex_replace(stripped_html,
                                  '[%s]' % re.escape(string.punctua
tion),
                                  '')
```

In [69]:
```
max_features = 10000
sequence_length = 250

vectorize_layer = TextVectorization(
    standardize=custom_standardization,
    max_tokens=max_features,
    output_mode='int',
    output_sequence_length=sequence_length)
```

In [70]:
```
# Make a text-only dataset (without labels), then call adapt
train_text = raw_train_ds.map(lambda x, y: x)
vectorize_layer.adapt(train_text)
```

In [71]:
```
def vectorize_text(text, label):
  text = tf.expand_dims(text, -1)
  return vectorize_layer(text), label
```

In [72]:
```
# retrieve a batch (of 32 reviews and labels) from the dataset
text_batch, label_batch = next(iter(raw_train_ds))
first_review, first_label = text_batch[0], label_batch[0]
print("Review", first_review)
print("Label", raw_train_ds.class_names[first_label])
print("Vectorized review", vectorize_text(first_review, first_label
))
```

```
Review tf.Tensor(b'Silent Night, Deadly Night 5 is the very last o
f the series, and like part 4, it\'s unrelated to the first three
except by title and the fact that it\'s a Christmas-themed horror
flick.<br /><br />Except to the oblivious, there\'s some obvious t
hings going on here...Mickey Rooney plays a toymaker named Joe Pet
to and his creepy son\'s name is Pino. Ring a bell, anyone? Now, a
little boy named Derek heard a knock at the door one evening, and
opened it to find a present on the doorstep for him. Even though i
t said "don\'t open till Christmas", he begins to open it anyway b
ut is stopped by his dad, who scolds him and sends him to bed, and
opens the gift himself. Inside is a little red ball that sprouts S
anta arms and a head, and proceeds to kill dad. Oops, maybe he sho
uld have left well-enough alone. Of course Derek is then traumatiz
ed by the incident since he watched it from the stairs, but he doe
sn\'t grow up to be some killer Santa, he just stops talking.<br /
><br />There\'s a mysterious stranger lurking around, who seems ve
ry interested in the toys that Joe Petto makes. We even see him bu
ying a bunch when Derek\'s mom takes him to the store to find a gi
ft for him to bring him out of his trauma. And what exactly is thi
s guy doing? Well, we\'re not sure but he does seem to be taking t
hese toys apart to see what makes them tick. He does keep his land
lord from evicting him by promising him to pay him in cash the nex
t day and presents him with a "Larry the Larvae" toy for his kid,
but of course "Larry" is not a good toy and gets out of the box in
the car and of course, well, things aren\'t pretty.<br /><br />Any
way, eventually what\'s going on with Joe Petto and Pino is of cou
rse revealed, and as with the old story, Pino is not a "real boy".
Pino is probably even more agitated and naughty because he suffers
from "Kenitalia" (a smooth plastic crotch) so that could account f
or his evil ways. And the identity of the lurking stranger is reve
aled too, and there\'s even kind of a happy ending of sorts. Whee.
<br /><br />A step up from part 4, but not much of one. Again, Bri
an Yuzna is involved, and Screaming Mad George, so some decent spe
cial effects, but not enough to make this great. A few leftovers f
rom part 4 are hanging around too, like Clint Howard and Neith Hun
ter, but that doesn\'t really make any difference. Anyway, I now h
ave seeing the whole series out of my system. Now if I could get s
ome of it out of my brain. 4 out of 5.', shape=(), dtype=string)
Label neg
Vectorized review (<tf.Tensor: shape=(1, 250), dtype=int64, numpy=
array([[1287,  313, 2380,  313,  661,    7,    2,   52,  229,    5
,    2,
         200,    3,   38,  170,  669,   29, 5492,    6,    2,   83
,  297,
         549,   32,  410,    3,    2,  186,   12,   29,    4,    1
```

```
,  191,
          510,  549,    6,    2, 8229,  212,   46,  576,  175,  168
,   20,
            1, 5361,  290,    4,    1,  761,  969,    1,    3,   24
,  935,
         2271,  393,    7,    1, 1675,    4, 3747,  250,  148,    4
,  112,
          436,  761, 3529,  548,    4, 3633,   31,    2, 1331,   28
, 2096,
            3, 2912,    9,    6,  163,    4, 1006,   20,    2,    1
,   15,
           85,   53,  147,    9,  292,   89,  959, 2314,  984,   27
,  762,
            6,  959,    9,  564,   18,    7, 2140,   32,   24, 1254
,   36,
            1,   85,    3, 3298,   85,    6, 1410,    3, 1936,    2
, 3408,
          301,  965,    7,    4,  112,  740, 1977,   12,    1, 2014
, 2772,
            3,    4,  428,    3, 5177,    6,  512, 1254,    1,  278
,   27,
          139,   25,  308,    1,  579,    5,  259, 3529,    7,   92
, 8981,
           32,    2, 3842,  230,   27,  289,    9,   35,    2, 5712
,   18,
           27,  144, 2166,   56,    6,   26,   46,  466, 2014,   27
,   40,
         2745,  657,  212,    4, 1376, 3002, 7080,  183,   36,  180
,   52,
          920,    8,    2, 4028,   12,  969,    1,  158,   71,   53
,   67,
           85, 2754,    4,  734,   51,    1, 1611,  294,   85,    6
,    2,
         1164,    6,  163,    4, 3408,   15,   85,    6,  717,   85
,   44,
            5,   24, 7158,    3,   48,  604,    7,   11,  225,  384
,   73,
           65,   21,  242,   18,   27,  120,  295,    6,   26,  667
,  129,
         4028,  948,    6,   67,   48,  158,   93,    1]])>, <tf.Te
nsor: shape=(), dtype=int32, numpy=0>)
```

In [73]:

```python
print("1287 ---> ",vectorize_layer.get_vocabulary()[1287])
print(" 313 ---> ",vectorize_layer.get_vocabulary()[313])
print('Vocabulary size: {}'.format(len(vectorize_layer.get_vocabula
ry())))
```

```
1287 --->  silent
 313 --->  night
Vocabulary size: 10000
```

In [74]:
```
train_ds = raw_train_ds.map(vectorize_text)
val_ds = raw_val_ds.map(vectorize_text)
test_ds = raw_test_ds.map(vectorize_text)
```

In [75]:
```
AUTOTUNE = tf.data.experimental.AUTOTUNE

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

In [76]:
```
embedding_dim = 16
```

In [77]:
```
model = tf.keras.Sequential([
  layers.Embedding(max_features + 1, embedding_dim),
  layers.Dropout(0.2),
  layers.GlobalAveragePooling1D(),
  layers.Dropout(0.2),
  layers.Dense(1)])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, None, 16)          160016    
_________________________________________________________________
dropout (Dropout)            (None, None, 16)          0         
_________________________________________________________________
global_average_pooling1d_1 ( (None, 16)                0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17        
=================================================================
Total params: 160,033
Trainable params: 160,033
Non-trainable params: 0
_________________________________________________________________
```

In [78]:
```
model.compile(loss=losses.BinaryCrossentropy(from_logits=True),
              optimizer='adam',
              metrics=tf.metrics.BinaryAccuracy(threshold=0.0))
```

In [79]:
```
epochs = 10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs)
```

```
Epoch 1/10
625/625 [==============================] - 11s 17ms/step - loss: 0
.6624 - binary_accuracy: 0.6930 - val_loss: 0.6124 - val_binary_ac
curacy: 0.7728
Epoch 2/10
625/625 [==============================] - 4s 7ms/step - loss: 0.5
471 - binary_accuracy: 0.8018 - val_loss: 0.4971 - val_binary_accu
racy: 0.8220
Epoch 3/10
625/625 [==============================] - 4s 6ms/step - loss: 0.4
436 - binary_accuracy: 0.8461 - val_loss: 0.4192 - val_binary_accu
racy: 0.8484
Epoch 4/10
625/625 [==============================] - 4s 6ms/step - loss: 0.3
772 - binary_accuracy: 0.8665 - val_loss: 0.3730 - val_binary_accu
racy: 0.8614
Epoch 5/10
625/625 [==============================] - 4s 6ms/step - loss: 0.3
342 - binary_accuracy: 0.8804 - val_loss: 0.3442 - val_binary_accu
racy: 0.8678
Epoch 6/10
625/625 [==============================] - 4s 7ms/step - loss: 0.3
042 - binary_accuracy: 0.8901 - val_loss: 0.3251 - val_binary_accu
racy: 0.8728
Epoch 7/10
625/625 [==============================] - 4s 7ms/step - loss: 0.2
800 - binary_accuracy: 0.8978 - val_loss: 0.3120 - val_binary_accu
racy: 0.8742
Epoch 8/10
625/625 [==============================] - 4s 6ms/step - loss: 0.2
600 - binary_accuracy: 0.9052 - val_loss: 0.3026 - val_binary_accu
racy: 0.8760
Epoch 9/10
625/625 [==============================] - 4s 6ms/step - loss: 0.2
451 - binary_accuracy: 0.9112 - val_loss: 0.2960 - val_binary_accu
racy: 0.8768
Epoch 10/10
625/625 [==============================] - 4s 6ms/step - loss: 0.2
314 - binary_accuracy: 0.9159 - val_loss: 0.2914 - val_binary_accu
racy: 0.8786
```

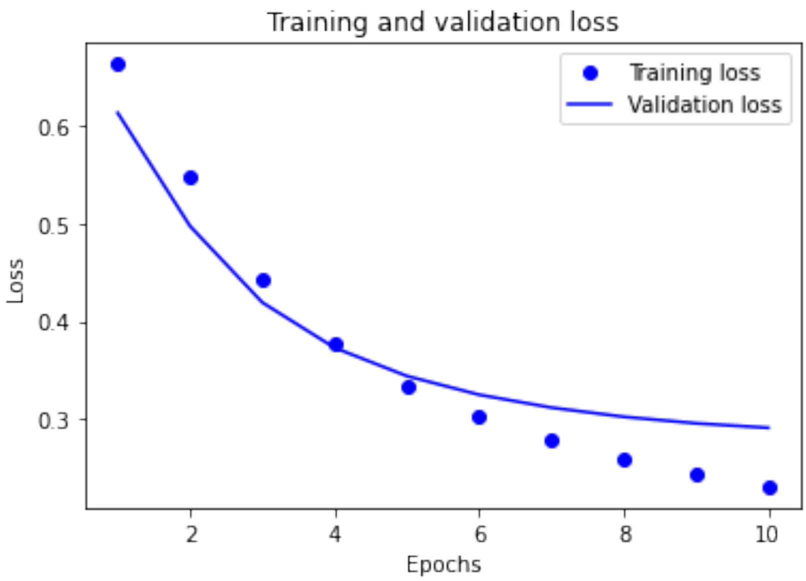
In [80]:
```
loss, accuracy = model.evaluate(test_ds)

print("Loss: ", loss)
print("Accuracy: ", accuracy)
```

```
782/782 [==============================] - 9s 12ms/step - loss: 0.
3099 - binary_accuracy: 0.8743
Loss:  0.30989280343055725
Accuracy:  0.8743199706077576
```

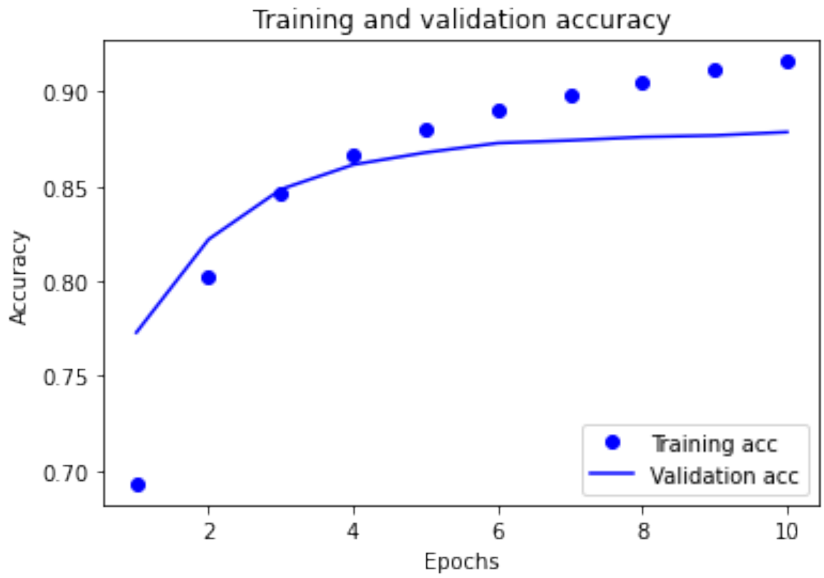
In [81]:
```
history_dict = history.history
history_dict.keys()
```

Out[81]:
```
dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accu
racy'])
```

In [82]:
```
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)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', 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.show()
```



In [83]:
```
plt.plot(epochs, acc, 'bo', 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')

plt.show()
```



In [84]:
```
export_model = tf.keras.Sequential([
  vectorize_layer,
  model,
  layers.Activation('sigmoid')
])

export_model.compile(
    loss=losses.BinaryCrossentropy(from_logits=False), optimizer="a
dam", metrics=['accuracy']
)

# Test it with `raw_test_ds`, which yields raw strings
loss, accuracy = export_model.evaluate(raw_test_ds)
print(accuracy)
```

```
782/782 [==============================] - 10s 13ms/step - loss: 0
.3099 - accuracy: 0.8743
0.8743199706077576
```

Testing on Random examples

In [85]:
```
examples = [
  "The movie was great!",
  "The movie was okay.",
  "The movie was terrible..."
]

export_model.predict(examples)
```

Out[85]:
```
array([[0.6330596],
       [0.4555972],
       [0.3716082]], dtype=float32)
```

In [ ]: