```
import tensorflow_datasets as tfds
import tensorflow as tf
```

The IMDB movie reviews dataset comes packaged in `tfds`. It has already been preprocessed so that the reviews (sequences of words) have been converted to sequences of integers, where each integer represents a specific word in a dictionary.

```
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

Downloading and preparing dataset imdb_reviews/subwords8k/1.0.0 (download: 80.23 MiB, ge

Dl Completed...: 100%	1/1 [00:03<00:00, 3.11s/ url]
Dl Size...: 100%	80/80 [00:03<00:00, 25.98 MiB/s]
	25000/0 [00:35<00:00, 838.81 examples/s]

```
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/subwords8k/1.0
```

50%	12431/25000 [00:00<00:00, 124307.75 examples/s]
	25000/0 [00:33<00:00, 916.86 examples/s]

```
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/subwords8k/1.0
```

55%	13808/25000 [00:00<00:00, 138077.17 examples/s]
	50000/0 [01:01<00:00, 869.27 examples/s]

```
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/subwords8k/1.0
```

68%	33766/50000 [00:00<20:20, 13.30 examples/s]

Dataset imdb_reviews downloaded and prepared to /root/tensorflow_datasets/imdb_reviews/s

```
train_examples_batch, train_labels_batch = next(iter(train_dataset))
print(train_examples_batch)
print(train_labels_batch)
```

```
tf.Tensor(
[  62   18   41  604  927   65    3  644 7968   21   35 5096   36   11
   43 2948 5240  102   50  681 7862 1244    3 3266   29  122  640    2
   26   14  279  438   35   79  349  384   11 1991    3  492   79  122
  188  117   33 4047 4531   14   65 7968    8 1819 3947    3   62   27
    9   41  577 5044 2629 2552 7193 7961 3642    3   19  107 3903  225
   85  198   72    1 1512  738 2347  102 6245    8   85  308   79 6936
 7961   23 4981 8044    3 6429 7961 1141 1335 1848 4848   55 3601 4217
 8050    2    5   59 3831 1484 8040 7974  174 5773   22 5240  102   18
  247   26    4 3903 1612 3902  291   11    4   27   13   18 4092 4008
```

```
 7961    6  119  213 2774    3   12  258 2306   13   91   29  171   52
  229    2 1245 5790  995 7968    8   52 2948 5240 8039 7968    8   74
 1249    3   12  117 2438 1369  192   39 7975], shape=(163,), dtype=int64)
tf.Tensor(0, shape=(), dtype=int64)
```

Text Encoding

The dataset info includes the encoder (a `tfds.features.text.SubwordTextEncoder`). This text encoder will reversibly encode any string, falling back to byte-encoding if necessary.

```
encoder = info.features['text'].encoder
print('Vocabulary size: {}'.format(encoder.vocab_size))
```

```
Vocabulary size: 8185
```

```
sample_string = 'Hello TensorFlow.'

encoded_string = encoder.encode(sample_string)
print('Encoded string is {}'.format(encoded_string))

original_string = encoder.decode(encoded_string)
print('The original string: "{}"'.format(original_string))
```

```
Encoded string is [4025, 222, 6307, 2327, 4043, 2120, 7975]
The original string: "Hello TensorFlow."
```

```
assert original_string == sample_string
for index in encoded_string:
  print('{} ----> {}'.format(index, encoder.decode([index])))
```

```
4025 ----> Hell
222 ----> o
6307 ----> Ten
2327 ----> sor
4043 ----> Fl
2120 ----> ow
7975 ----> .
```

Create batches of training data for your model. The reviews are all different lengths, so use `padded_batch` to zero pad the sequences while batching.

```
BUFFER_SIZE = 10000
BATCH_SIZE = 64

train_dataset = train_dataset.shuffle(BUFFER_SIZE)
train_dataset = train_dataset.padded_batch(BATCH_SIZE)

test_dataset = test_dataset.padded_batch(BATCH_SIZE)
```

Build Model with an LSTM layer

Creating a `tf.keras.Sequential` model and start with an embedding layer. An embedding layer stores one vector per word. When called, it converts the sequences of word indices to sequences of vectors. These vectors are trainable. After training (on enough data), words with similar meanings often have similar vectors.

This index-lookup is much more efficient than the equivalent operation of passing a one-hot encoded vector through a `tf.keras.layers.Dense layer.`

A recurrent neural network (RNN) processes sequence input by iterating through the elements. RNNs pass the outputs from one timestep to their input—and then to the next.

The `tf.keras.layers.Bidirectional` wrapper can also be used with an RNN layer. This propagates the input forward and backwards through the RNN layer and then concatenates the output. This helps the RNN to learn long range dependencies. Keras sequential model here since all the layers in the model only have single input and produce single output.

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(encoder.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 64)          523840
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               66048
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 598,209
Trainable params: 598,209
Non-trainable params: 0
_________________________________________________________________
```

Since this is a binary classification problem and the model outputs logits (a single-unit layer with a linear activation), we'll use the `binary_crossentropy` loss function. It is better for dealing with probabilities—it measures the "distance" between probability distributions, or in our case, between

```
model.compile(loss = tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer = tf.keras.optimizers.Adam(1e-4),
              metrics = ['accuracy'])
```

Train the model for 10 `epochs`. This is 10 iterations over all samples in the `train_dataset` tensors.

```
history = model.fit(train_dataset, epochs = 10, validation_data = test_dataset, validation_st
```

```
Epoch 1/10
391/391 [==============================] - 817s 2s/step - loss: 0.6397 - accuracy: 0.57
Epoch 2/10
391/391 [==============================] - 867s 2s/step - loss: 0.3491 - accuracy: 0.85
Epoch 3/10
391/391 [==============================] - 859s 2s/step - loss: 0.2509 - accuracy: 0.90
Epoch 4/10
391/391 [==============================] - 861s 2s/step - loss: 0.2078 - accuracy: 0.92
Epoch 5/10
391/391 [==============================] - 862s 2s/step - loss: 0.1843 - accuracy: 0.93
Epoch 6/10
391/391 [==============================] - 863s 2s/step - loss: 0.1591 - accuracy: 0.94
Epoch 7/10
391/391 [==============================] - 863s 2s/step - loss: 0.1521 - accuracy: 0.94
Epoch 8/10
391/391 [==============================] - 864s 2s/step - loss: 0.1359 - accuracy: 0.95
Epoch 9/10
391/391 [==============================] - 869s 2s/step - loss: 0.1267 - accuracy: 0.95
Epoch 10/10
391/391 [==============================] - 864s 2s/step - loss: 0.1137 - accuracy: 0.96
```

Test For Accuracy

```
test_loss, test_acc = model.evaluate(test_dataset)

print('Test Loss: {}'.format(test_loss))
print('Test Accuracy: {}'.format(test_acc))
```

```
391/391 [==============================] - 183s 468ms/step - loss: 0.4283 - accuracy: 0
Test Loss: 0.4282757043838501
Test Accuracy: 0.854960024356842
```

Prediction Functions

```
def pad_to_size(vec, size):
  zeros = [0] * (size - len(vec))
```

```
    vec.extend(zeros)
    return vec


def sample_predict(sample_pred_text, pad):
  encoded_sample_pred_text = encoder.encode(sample_pred_text)

  if pad:
    encoded_sample_pred_text = pad_to_size(encoded_sample_pred_text, 64)
  encoded_sample_pred_text = tf.cast(encoded_sample_pred_text, tf.float32)
  predictions = model.predict(tf.expand_dims(encoded_sample_pred_text, 0))
  print("Prediction Score: ", predictions)

  output = ""
  if predictions[0][0] >= 0.5: output = "POSITIVE"
  elif predictions[0][0] <= -1: output = "NEGATIVE"
  else: output = "NEUTRAL"

  return output
```

Prediction with Sample Sentiments

```
sample_pred_text = ('The movie was not good. The animation and the graphics were terrible. I
predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
Prediction Score:  [[-1.2465143]]
NEGATIVE
```

```
sample_pred_text = ('The movie was cool. The animation and the graphics were out of this worl
predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
Prediction Score:  [[0.11379167]]
NEUTRAL
```

```
sample_pred_text = ('This movie is awesome. The acting was incredicable. Highly recommend')
predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
Prediction Score:  [[1.4532732]]
POSITIVE
```

```
sample_pred_text = ('This movie was so so. The acting was medicore. Kind of recommend')
predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
Prediction Score:  [[-0.49171886]]
NEUTRAL
```

```
# AVENGERS: ENDGAME 5 STAR COMMENT

sample_pred_text = ("""I loved the movie a lot as I am great fan of marvel! Avengers: Endgame
and surely an enthralling experience. The last film of the 'Avengers' franchise is remarkable
in one film is just surpassing. Marvel has been working on this grand culmination ever since
work and ambition has paid off. The directors, Anthony and Joe Russo, have made sure that it
Stephen McFeely have come up with a screenplay full of epic and unpredictable moments. The fi
The biggest strength of the film is the emotions. This is the most emotional superhero film I
were jaw-dropping. The climatic battle left me amazed. It's just filled with memorable moment
and have a great impact on the film. The humour doesn't look exaggerated and manages to enter
and suspenseful. The film features many cameos of characters from the previous MCU films, whi
gives me goosebumps, though I've listened to it several times. It was really clever to make c
But the show-stealer is Robert Downey Jr, who plays the role of Tony Stark/Iron Man. The man
can perfect his role. Do not miss his powerful moments in the final battle.""")

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<
Prediction Score:  [[7.5766687]]
POSITIVE
```

```
# AVENGERS: ENDGAME 3 STAR COMMENT

sample_pred_text = ("""Overrated Sequel, But Still Good, But Violent! Beloved characters die,
stabbings, punching, shooting, and more. The characters swear a bit. Even Captain America doe
Black Widow, Hawkeye. Thor not so much because he SPOILER ALERT: got fat and played Fortnite
kid wanting to say 'I want to be like the God of Thunder and play fortnite all day'. Characte
himself of something but not saying what. With reviewing the movie, the first half hour was g
pretty good, but the last hour was epic. From just starting out with Iron Man, Cap, Thor, and
really pulled it off. Overall, pretty good for families and a good finale for the Infinity Sa

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<
Prediction Score:  [[-3.3150415]]
NEGATIVE
```

```
# AVENGERS: ENDGAME 3 STAR COMMENT

sample_pred_text = ("""I don't get why so many people like this movie so much, all they did w
They also just added a whole bunch of scenes of previous marvel movies, and that is how they
Now, getting to the inappropiate content for the parents. This is just your average superhero
There are also strong roll models, but if you have a kid that is in elementary school or high
But, I am not trying to parent your child, I am just giving my personal opinion, so you can c
```

```
But, I am not trying to parent your child, I am just giving my personal opinion, so you can c
watching it.""")

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:7 out of the last 7 calls to <function Model.make_predict_function.<l
WARNING:tensorflow:7 out of the last 7 calls to <function Model.make_predict_function.<l
Prediction Score:  [[-2.6585133]]
NEGATIVE
```

```
# AVENGERS: ENDGAME 1 STAR COMMENT

sample_pred_text = ("""Disappointing storyline - too many sad crying scenes - too much shit s
an adult I don't appreciate swearing in movies. There are MANY people who don't use cuss word
To hear Robert Downey jr's 'moment' with his young child using and laughing at the fact that 
ages watching this, that it is okay when it isn't. They seemed to want to use their cuss word
how they get their best laughs from audience. Bring back your creative, quirky writers from t
development and writing without resorting to desperate shock value. We have loved every movie
was soooo boring in this. All of us watching kept hoping it would improve and it didn't. I th
between Quill (StarLord) and Thor. Subtle, but funny. We all could've cared less if anyone di
Why! It was torture and I felt robbed of my time in the end.""")

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:8 out of the last 8 calls to <function Model.make_predict_function.<l
WARNING:tensorflow:8 out of the last 8 calls to <function Model.make_predict_function.<l
Prediction Score:  [[-2.4277794]]
NEGATIVE
```

```
# AVENGERS ENDGAME COMMENT

sample_pred_text = ("""What a great way to end several major storylines that they invested in
I feel like this is just the cherry on the top. My only complaint is something that you can't
'powers' are totally inconsistent from scene to scene, and movie to movie. This is a trope th
Captain Marvel, Thor, Scarlet Witch, and Hulk were always as powerful as they show flashes of
could destroy Thanos in the blink of an eye, and have done similar feats in other stories (an
sometimes 'reduce' their power to a lower level, without an explained mechanism, is pretty la
Yes, this constant Ex Machina is needed to maintain the drama and keep the plot going, but it

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:9 out of the last 9 calls to <function Model.make_predict_function.<l
WARNING:tensorflow:9 out of the last 9 calls to <function Model.make_predict_function.<l
Prediction Score:  [[3.6573927]]
POSITIVE
```

```
sample_pred_text = ("""Superhero comics, and much of their adaptations, have long taken an ou
At their best, they can take these fantastical ideas and make them emotionally resonant, even
In some respects, Endgame pulls this off beautifully, like how the character Nebula confronts
her personal growth. But as fun as the movie is, there's an undeniable hollowness at its core
ideas and symbols.""")

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:10 out of the last 10 calls to <function Model.make_predict_function
WARNING:tensorflow:10 out of the last 10 calls to <function Model.make_predict_function
Prediction Score:  [[1.5968482]]
POSITIVE
```

```
# AUGUST to JUNE: Bringing Life to School (http://augusttojune.com/press-media/audience-comme

sample_pred_text = ("""The film's ever-present focus on the 'big picture' of education and li
In particular, we really liked seeing how you conferenced with parents, e.g., paraphrased:  '
readers and late readers were', the overall approach to literacy- holistically focused rather
growth/experiences/conflict resolution, and the fact that you did not choose to hide those mo
children when disruptive.  There is a whole, whole, whole lot more that I wish to say! It is
heart-warming to know that your school carries on.""")

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

```
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function
Prediction Score:  [[1.7408558]]
POSITIVE
```

```
# AUGUST to JUNE: Bringing Life to School

sample_pred_text = ("""After we screened August To June, Boynton, Boca Democratic Party Movie
they never seen an audience response so serene. People wanted to stay. No one was angry. Conv
who had known each other for years discovered commonalities that previously they did not know
The response, in unison, August To June is warm.  It touches people. Real life school situati
already is. People were reminded of good times and the challenges that helped them grow great
reverence.""")

predictions = sample_predict(sample_pred_text, pad = False)
print(predictions)
```

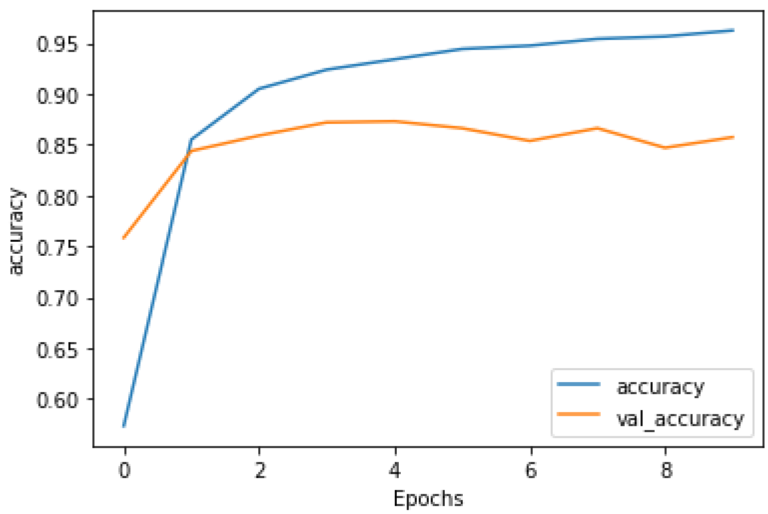
```
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function
Prediction Score:  [[4.970427]]
POSITIVE
```

Plotting Acuracy & Loss Function Graphs

```
import matplotlib.pyplot as plt

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

plot_graphs(history, 'accuracy')
```




```
plot_graphs(history, 'loss')
```