```
!curl -O https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!tar -xf aclImdb_v1.tar.gz
!rm -r aclImdb/train/unsup
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 80.2M  100 80.2M    0     0  55.1M      0  0:00:01  0:00:01 --:--:-- 55.1M
```

Preparing the data

```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_dir = base_dir / "train"
for category in ("neg", "pos"):
    os.makedirs(val_dir / category)
    files = os.listdir(train_dir / category)
    random.Random(1337).shuffle(files)
    num_val_samples = int(0.2 * len(files))
    val_files = files[-num_val_samples:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)

train_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/train", batch_size=batch_size
)
val_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/val", batch_size=batch_size
)
test_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/test", batch_size=batch_size
)
text_only_train_ds = train_ds.map(lambda x, y: x)
```

```
Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

Preparing integer sequence datasets

```
from tensorflow.keras import layers

max_length = 600
max_tokens = 20000
text_vectorization = layers.TextVectorization(
    max_tokens=max_tokens,
    output_mode="int",
    output_sequence_length=max_length,
)
text_vectorization.adapt(text_only_train_ds)

int_train_ds = train_ds.map(
    lambda x, y: (text_vectorization(x), y),
    num_parallel_calls=4)
int_val_ds = val_ds.map(
    lambda x, y: (text_vectorization(x), y),
    num_parallel_calls=4)
int_test_ds = test_ds.map(
    lambda x, y: (text_vectorization(x), y),
    num_parallel_calls=4)
```

A sequence model built on one-hot encoded vector sequences

```
import tensorflow as tf
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = tf.one_hot(inputs, depth=max_tokens)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None)]            0

 tf.one_hot (TFOpLambda)     (None, None, 20000)       0

 bidirectional (Bidirectiona  (None, 64)               5128448
 l)

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 5,128,513
Trainable params: 5,128,513
Non-trainable params: 0
_________________________________________________________________
```

Training a first basic sequence model

```
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm.keras",
                                    save_best_only=True)
]
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10, callbacks=callbacks)
model = keras.models.load_model("one_hot_bidir_lstm.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Epoch 1/10
625/625 [==============================] - 179s 274ms/step - loss: 0.5486 - accuracy: 0.7329 - val_loss: 0.3650 - val_accuracy: 0.8620
Epoch 2/10
625/625 [==============================] - 173s 277ms/step - loss: 0.3683 - accuracy: 0.8660 - val_loss: 0.3805 - val_accuracy: 0.8612
Epoch 3/10
625/625 [==============================] - 174s 278ms/step - loss: 0.2886 - accuracy: 0.8969 - val_loss: 0.3146 - val_accuracy: 0.8702
Epoch 4/10
625/625 [==============================] - 173s 277ms/step - loss: 0.2411 - accuracy: 0.9169 - val_loss: 0.2856 - val_accuracy: 0.8836
Epoch 5/10
625/625 [==============================] - 173s 277ms/step - loss: 0.1999 - accuracy: 0.9320 - val_loss: 0.2937 - val_accuracy: 0.8858
Epoch 6/10
625/625 [==============================] - 173s 276ms/step - loss: 0.1886 - accuracy: 0.9381 - val_loss: 0.3439 - val_accuracy: 0.8770
Epoch 7/10
625/625 [==============================] - 174s 278ms/step - loss: 0.1649 - accuracy: 0.9437 - val_loss: 0.3413 - val_accuracy: 0.8802
Epoch 8/10
625/625 [==============================] - 174s 279ms/step - loss: 0.1481 - accuracy: 0.9525 - val_loss: 0.3170 - val_accuracy: 0.8814
Epoch 9/10
625/625 [==============================] - 173s 276ms/step - loss: 0.1207 - accuracy: 0.9599 - val_loss: 0.3794 - val_accuracy: 0.8820
Epoch 10/10
625/625 [==============================] - 173s 277ms/step - loss: 0.1041 - accuracy: 0.9665 - val_loss: 0.3643 - val_accuracy: 0.8808

782/782 [==============================] - 102s 129ms/step - loss: 0.3198 - accuracy: 0.8682
Test acc: 0.868
```

Understanding word embeddings

Learning word embeddings with the Embedding layer

Instantiating an `Embedding` layer

```
embedding_layer = layers.Embedding(input_dim=max_tokens, output_dim=256)
```

Model that uses an `Embedding` layer trained from scratch

```
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = layers.Embedding(input_dim=max_tokens, output_dim=256)(inputs)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()

callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru.keras",
                                    save_best_only=True)
]
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10, callbacks=callbacks)
model = keras.models.load_model("embeddings_bidir_gru.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, None)]            0

 embedding_1 (Embedding)     (None, None, 256)         5120000

 bidirectional_1 (Bidirectio  (None, 64)               73984
 nal)

 dropout_1 (Dropout)         (None, 64)                0
```

```
 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 5,194,049
Trainable params: 5,194,049
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
625/625 [==============================] - 38s 56ms/step - loss: 0.4530 - accuracy: 0.8001 - val_loss: 0.3148 - val_accuracy: 0.8772
Epoch 2/10
625/625 [==============================] - 35s 56ms/step - loss: 0.2915 - accuracy: 0.8919 - val_loss: 0.3256 - val_accuracy: 0.8646
Epoch 3/10
625/625 [==============================] - 34s 54ms/step - loss: 0.2360 - accuracy: 0.9176 - val_loss: 0.3458 - val_accuracy: 0.8752
Epoch 4/10
625/625 [==============================] - 37s 58ms/step - loss: 0.1987 - accuracy: 0.9323 - val_loss: 0.4247 - val_accuracy: 0.8550
Epoch 5/10
625/625 [==============================] - 35s 56ms/step - loss: 0.1692 - accuracy: 0.9434 - val_loss: 0.3369 - val_accuracy: 0.8506
Epoch 6/10
625/625 [==============================] - 35s 56ms/step - loss: 0.1411 - accuracy: 0.9534 - val_loss: 0.4381 - val_accuracy: 0.8542
Epoch 7/10
625/625 [==============================] - 35s 55ms/step - loss: 0.1219 - accuracy: 0.9615 - val_loss: 0.3460 - val_accuracy: 0.8802
Epoch 8/10
625/625 [==============================] - 34s 54ms/step - loss: 0.1041 - accuracy: 0.9686 - val_loss: 0.4552 - val_accuracy: 0.8608
Epoch 9/10
625/625 [==============================] - 34s 54ms/step - loss: 0.0828 - accuracy: 0.9740 - val_loss: 0.4044 - val_accuracy: 0.8742
Epoch 10/10
625/625 [==============================] - 36s 58ms/step - loss: 0.0712 - accuracy: 0.9790 - val_loss: 0.4480 - val_accuracy: 0.8744
782/782 [==============================] - 27s 34ms/step - loss: 0.3339 - accuracy: 0.8662
Test acc: 0.866
```

Understanding padding and masking

Using an `Embedding` layer with masking enabled

```
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = layers.Embedding(
    input_dim=max_tokens, output_dim=256, mask_zero=True)(inputs)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()

callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru_with_masking.keras",
                                    save_best_only=True)
]
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10, callbacks=callbacks)
model = keras.models.load_model("embeddings_bidir_gru_with_masking.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, None)]            0

 embedding_2 (Embedding)     (None, None, 256)         5120000

 bidirectional_2 (Bidirectio  (None, 64)               73984
 nal)

 dropout_2 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 5,194,049
Trainable params: 5,194,049
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
625/625 [==============================] - 43s 59ms/step - loss: 0.3908 - accuracy: 0.8263 - val_loss: 0.2684 - val_accuracy: 0.8930
Epoch 2/10
625/625 [==============================] - 35s 56ms/step - loss: 0.2197 - accuracy: 0.9161 - val_loss: 0.2691 - val_accuracy: 0.8884
Epoch 3/10
625/625 [==============================] - 35s 56ms/step - loss: 0.1698 - accuracy: 0.9369 - val_loss: 0.2999 - val_accuracy: 0.8752
Epoch 4/10
625/625 [==============================] - 35s 56ms/step - loss: 0.1262 - accuracy: 0.9549 - val_loss: 0.3223 - val_accuracy: 0.8844
Epoch 5/10
625/625 [==============================] - 35s 56ms/step - loss: 0.0940 - accuracy: 0.9672 - val_loss: 0.3332 - val_accuracy: 0.8852
Epoch 6/10
625/625 [==============================] - 38s 61ms/step - loss: 0.0689 - accuracy: 0.9763 - val_loss: 0.3876 - val_accuracy: 0.8820
Epoch 7/10
625/625 [==============================] - 37s 58ms/step - loss: 0.0492 - accuracy: 0.9833 - val_loss: 0.4360 - val_accuracy: 0.8636
Epoch 8/10
625/625 [==============================] - 38s 61ms/step - loss: 0.0344 - accuracy: 0.9886 - val_loss: 0.5016 - val_accuracy: 0.8648
Epoch 9/10
625/625 [==============================] - 35s 56ms/step - loss: 0.0212 - accuracy: 0.9934 - val_loss: 0.5159 - val_accuracy: 0.8750
Epoch 10/10
625/625 [==============================] - 36s 57ms/step - loss: 0.0178 - accuracy: 0.9942 - val_loss: 0.5390 - val_accuracy: 0.8700
782/782 [==============================] - 24s 27ms/step - loss: 0.2779 - accuracy: 0.8856
Test acc: 0.886
```

Using pretrained word embeddings

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip -q glove.6B.zip
```

```
--2022-12-10 11:09:04--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2022-12-10 11:09:04--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2022-12-10 11:09:04--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  5.02MB/s    in 2m 39s

2022-12-10 11:11:43 (5.17 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

Parsing the GloVe word-embeddings file

```
import numpy as np
path_to_glove_file = "glove.6B.100d.txt"

embeddings_index = {}
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs

print(f"Found {len(embeddings_index)} word vectors.")
```

```
Found 400000 word vectors.
```

Preparing the GloVe word-embeddings matrix

```
embedding_dim = 100

vocabulary = text_vectorization.get_vocabulary()
word_index = dict(zip(vocabulary, range(len(vocabulary))))

embedding_matrix = np.zeros((max_tokens, embedding_dim))
for word, i in word_index.items():
    if i < max_tokens:
        embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
embedding_layer = layers.Embedding(
    max_tokens,
    embedding_dim,
    embeddings_initializer=keras.initializers.Constant(embedding_matrix),
    trainable=False,
    mask_zero=True,
)
```

Model that uses a pretrained Embedding layer

```
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = embedding_layer(inputs)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()

callbacks = [
    keras.callbacks.ModelCheckpoint("glove_embeddings_sequence_model.keras",
                                    save_best_only=True)
]
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10, callbacks=callbacks)
model = keras.models.load_model("glove_embeddings_sequence_model.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, None)]            0

 embedding_3 (Embedding)     (None, None, 100)         2000000

 bidirectional_3 (Bidirectio  (None, 64)               34048
 nal)

 dropout_3 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 2,034,113
Trainable params: 34,113
Non-trainable params: 2,000,000
_________________________________________________________________
Epoch 1/10
625/625 [==============================] - 42s 55ms/step - loss: 0.5771 - accuracy: 0.6956 - val_loss: 0.4965 - val_accuracy: 0.7640
Epoch 2/10
625/625 [==============================] - 33s 53ms/step - loss: 0.4500 - accuracy: 0.7944 - val_loss: 0.3971 - val_accuracy: 0.8278
Epoch 3/10
625/625 [==============================] - 34s 55ms/step - loss: 0.4029 - accuracy: 0.8242 - val_loss: 0.3897 - val_accuracy: 0.8270
Epoch 4/10
625/625 [==============================] - 33s 53ms/step - loss: 0.3720 - accuracy: 0.8367 - val_loss: 0.3488 - val_accuracy: 0.8440
Epoch 5/10
625/625 [==============================] - 33s 52ms/step - loss: 0.3476 - accuracy: 0.8523 - val_loss: 0.3542 - val_accuracy: 0.8498
Epoch 6/10
625/625 [==============================] - 32s 52ms/step - loss: 0.3265 - accuracy: 0.8633 - val loss: 0.3237 - val accuracy: 0.8604
```

```
Epoch 7/10
625/625 [==============================] - 31s 50ms/step - loss: 0.3065 - accuracy: 0.8726 - val_loss: 0.3255 - val_accuracy: 0.8550
Epoch 8/10
625/625 [==============================] - 33s 52ms/step - loss: 0.2920 - accuracy: 0.8799 - val_loss: 0.3188 - val_accuracy: 0.8710
Epoch 9/10
625/625 [==============================] - 33s 52ms/step - loss: 0.2730 - accuracy: 0.8899 - val_loss: 0.2982 - val_accuracy: 0.8752
Epoch 10/10
625/625 [==============================] - 33s 53ms/step - loss: 0.2622 - accuracy: 0.8951 - val_loss: 0.3050 - val_accuracy: 0.8680
782/782 [==============================] - 34s 41ms/step - loss: 0.3010 - accuracy: 0.8719
Test acc: 0.872
```