AML_4_bchennu (1)

July 28, 2024

Downloading IMDB Dataset

[ ]:

```
!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  49.3M      0  0:00:01  0:00:01 --:--:-- 49.3M
```

Preparing the data

[ ]:

```
shutil.rmtree('aclImdb/val')
```

[ ]:

```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_n = base_dir / "train_n"
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 = 10000
    val_files = files[-num_val_samples:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)
```

[ ]:

```
shutil.rmtree('aclImdb/train_n')
```

1 Training Sample Size: 100

[ ]:

```
for category in ("neg", "pos"):
    os.makedirs(train_n / category)
    files = os.listdir(train_dir / category)
    num_train_samples=100
    train_files = files[-num_train_samples:]
    for fname in train_files:
        shutil.move(train_dir / category / fname,
                    train_n / category / fname)

train_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/train_n", 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 200 files belonging to 2 classes.
Found 20000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

Preparing integer sequence datasets

[ ]:

```
from tensorflow.keras import layers

max_length = 150
max_tokens = 10000
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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_17"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_18 (InputLayer)       [(None, None)]            0

 tf.one_hot_4 (TFOpLambda)   (None, None, 10000)       0

 bidirectional_17 (Bidirect  (None, 64)                2568448
 ional)

 dropout_17 (Dropout)        (None, 64)                0

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 2568513 (9.80 MB)
Trainable params: 2568513 (9.80 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training a first basic sequence model

[ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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
7/7 [==============================] - 12s 1s/step - loss: 1.8091 - accuracy:
```

```
0.5000 - val_loss: 1.0221 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 0.8707 - accuracy:
0.5000 - val_loss: 0.7331 - val_accuracy: 0.5000
Epoch 3/10
7/7 [==============================] - 8s 1s/step - loss: 0.7080 - accuracy:
0.6400 - val_loss: 0.7438 - val_accuracy: 0.5000
Epoch 4/10
7/7 [==============================] - 8s 1s/step - loss: 0.6363 - accuracy:
0.6200 - val_loss: 0.7506 - val_accuracy: 0.5000
Epoch 5/10
7/7 [==============================] - 8s 1s/step - loss: 0.5680 - accuracy:
0.7150 - val_loss: 0.6849 - val_accuracy: 0.5288
Epoch 6/10
7/7 [==============================] - 8s 1s/step - loss: 0.5226 - accuracy:
0.7450 - val_loss: 0.7532 - val_accuracy: 0.5000
Epoch 7/10
7/7 [==============================] - 8s 1s/step - loss: 0.4505 - accuracy:
0.8450 - val_loss: 0.6879 - val_accuracy: 0.5199
Epoch 8/10
7/7 [==============================] - 8s 1s/step - loss: 0.8693 - accuracy:
0.7550 - val_loss: 0.7041 - val_accuracy: 0.5081
Epoch 9/10
7/7 [==============================] - 8s 1s/step - loss: 0.2954 - accuracy:
0.9550 - val_loss: 0.6376 - val_accuracy: 0.6514
Epoch 10/10
7/7 [==============================] - 8s 1s/step - loss: 0.1882 - accuracy:
0.9650 - val_loss: 0.7546 - val_accuracy: 0.5155
782/782 [==============================] - 12s 13ms/step - loss: 0.6386 -
accuracy: 0.6538
Test acc: 0.654
```

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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_20 (InputLayer)       [(None, None)]            0

 embedding_12 (Embedding)    (None, None, 256)         2560000

 bidirectional_19 (Bidirect  (None, 64)                73984
 ional)

 dropout_19 (Dropout)        (None, 64)                0

 dense_19 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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}")
```

```
Epoch 1/10
7/7 [==============================] - 9s 911ms/step - loss: 1.6602 - accuracy:
0.5050 - val_loss: 0.8808 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 5s 787ms/step - loss: 0.7400 - accuracy:
0.5400 - val_loss: 0.7357 - val_accuracy: 0.5069
Epoch 3/10
7/7 [==============================] - 5s 764ms/step - loss: 0.6072 - accuracy:
0.6650 - val_loss: 0.7056 - val_accuracy: 0.5181
Epoch 4/10
7/7 [==============================] - 5s 738ms/step - loss: 0.5148 - accuracy:
0.7650 - val_loss: 0.6833 - val_accuracy: 0.5554
Epoch 5/10
7/7 [==============================] - 4s 723ms/step - loss: 0.4763 - accuracy:
0.8150 - val_loss: 0.7121 - val_accuracy: 0.5469
Epoch 6/10
7/7 [==============================] - 5s 791ms/step - loss: 0.2438 - accuracy:
```

```
0.9600 - val_loss: 0.7288 - val_accuracy: 0.5652
Epoch 7/10
7/7 [==============================] - 5s 775ms/step - loss: 0.1281 - accuracy:
0.9850 - val_loss: 0.7296 - val_accuracy: 0.6015
Epoch 8/10
7/7 [==============================] - 5s 742ms/step - loss: 0.0465 - accuracy:
0.9950 - val_loss: 0.9009 - val_accuracy: 0.6114
Epoch 9/10
7/7 [==============================] - 4s 722ms/step - loss: 0.0177 - accuracy:
1.0000 - val_loss: 1.6242 - val_accuracy: 0.6116
Epoch 10/10
7/7 [==============================] - 4s 715ms/step - loss: 0.0359 - accuracy:
0.9900 - val_loss: 1.0082 - val_accuracy: 0.5983
782/782 [==============================] - 7s 7ms/step - loss: 0.6827 -
accuracy: 0.5580
Test acc: 0.558
```

Understanding padding and masking **Using an Embedding layer with masking enabled**

[ ]:

```python
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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_20"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_21 (InputLayer)       [(None, None)]            0

 embedding_13 (Embedding)    (None, None, 256)         2560000

 bidirectional_20 (Bidirect  (None, 64)                73984
 ional)

 dropout_20 (Dropout)        (None, 64)                0

 dense_20 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
```

```
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```python
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}")
```

```
Epoch 1/10
7/7 [==============================] - 16s 1s/step - loss: 1.8803 - accuracy:
0.5000 - val_loss: 1.1719 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 5s 832ms/step - loss: 0.9612 - accuracy:
0.5000 - val_loss: 1.0232 - val_accuracy: 0.5000
Epoch 3/10
7/7 [==============================] - 5s 832ms/step - loss: 0.7860 - accuracy:
0.5000 - val_loss: 0.8841 - val_accuracy: 0.5000
Epoch 4/10
7/7 [==============================] - 5s 812ms/step - loss: 0.6177 - accuracy:
0.5400 - val_loss: 0.7503 - val_accuracy: 0.5078
Epoch 5/10
7/7 [==============================] - 5s 824ms/step - loss: 0.5158 - accuracy:
0.7300 - val_loss: 0.7387 - val_accuracy: 0.5120
Epoch 6/10
7/7 [==============================] - 5s 834ms/step - loss: 0.3970 - accuracy:
0.8800 - val_loss: 0.7187 - val_accuracy: 0.5251
Epoch 7/10
7/7 [==============================] - 5s 793ms/step - loss: 0.2462 - accuracy:
0.9750 - val_loss: 0.7244 - val_accuracy: 0.5343
Epoch 8/10
7/7 [==============================] - 5s 808ms/step - loss: 0.1459 - accuracy:
0.9850 - val_loss: 0.6855 - val_accuracy: 0.5971
Epoch 9/10
7/7 [==============================] - 5s 771ms/step - loss: 0.0562 - accuracy:
0.9950 - val_loss: 0.9840 - val_accuracy: 0.5170
Epoch 10/10
7/7 [==============================] - 5s 776ms/step - loss: 0.0187 - accuracy:
1.0000 - val_loss: 1.1697 - val_accuracy: 0.5159
782/782 [==============================] - 10s 7ms/step - loss: 0.6830 -
accuracy: 0.5974
Test acc: 0.597
```

Using pretrained word embeddings

[1]:

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

```
--2024-07-28 20:26:55--  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]
--2024-07-28 20:26:55--  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]
--2024-07-28 20:26:55--  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  4.77MB/s    in 2m 39s

2024-07-28 20:29:34 (5.18 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

Parsing the GloVe word-embeddings file

[ ]:

```python
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

[ ]:

```python
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="relu")(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,
                                    monitor="val_loss")
]
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_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_22 (InputLayer)       [(None, None)]            0

 embedding_10 (Embedding)    (None, None, 256)         2560000

 bidirectional_21 (Bidirect  (None, 64)                73984
 ional)

 dropout_21 (Dropout)        (None, 64)                0

 dense_21 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
7/7 [==============================] - 9s 900ms/step - loss: 1.8917 - accuracy:
0.5000 - val_loss: 0.8866 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 5s 798ms/step - loss: 0.7776 - accuracy:
0.5500 - val_loss: 0.7519 - val_accuracy: 0.5059
Epoch 3/10
7/7 [==============================] - 5s 794ms/step - loss: 0.6445 - accuracy:
0.5950 - val_loss: 0.7397 - val_accuracy: 0.5167
Epoch 4/10
7/7 [==============================] - 5s 735ms/step - loss: 0.5737 - accuracy:
0.6600 - val_loss: 0.7019 - val_accuracy: 0.5340
Epoch 5/10
7/7 [==============================] - 4s 717ms/step - loss: 0.4414 - accuracy:
0.8450 - val_loss: 0.7067 - val_accuracy: 0.5446
Epoch 6/10
7/7 [==============================] - 5s 753ms/step - loss: 0.2946 - accuracy:
0.9150 - val_loss: 0.8867 - val_accuracy: 0.5136
Epoch 7/10
7/7 [==============================] - 5s 761ms/step - loss: 0.1899 - accuracy:
0.9500 - val_loss: 0.9559 - val_accuracy: 0.5845
Epoch 8/10
7/7 [==============================] - 5s 738ms/step - loss: 0.0862 - accuracy:
0.9850 - val_loss: 1.0401 - val_accuracy: 0.5612
Epoch 9/10
7/7 [==============================] - 4s 706ms/step - loss: 0.0411 - accuracy:
0.9950 - val_loss: 0.8439 - val_accuracy: 0.5978
Epoch 10/10
7/7 [==============================] - 4s 706ms/step - loss: 0.0219 - accuracy:
1.0000 - val_loss: 1.3470 - val_accuracy: 0.5704
782/782 [==============================] - 7s 7ms/step - loss: 0.6998 -
accuracy: 0.5361
Test acc: 0.536
```

```
[ ]:
```

2 Training Sample Size: 800

[ ]:
```
!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  29.6M      0  0:00:02  0:00:02 --:--:-- 29.6M
```

[ ]:
```
shutil.rmtree('aclImdb/val')
```

[ ]:
```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_n = base_dir / "train_n"
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 = 10000
    val_files = files[-num_val_samples:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)
```

[ ]:
```
shutil.rmtree('aclImdb/train_n')
```

[ ]:
```
for category in ("neg", "pos"):
    os.makedirs(train_n / category)
    files = os.listdir(train_dir / category)
    num_train_samples=800
    train_files = files[-num_train_samples:]
    for fname in train_files:
        shutil.move(train_dir / category / fname,
                    train_n / category / fname)

train_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/train_n", 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 1600 files belonging to 2 classes.
Found 20000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

[ ]:

```
from tensorflow.keras import layers

max_length = 150
max_tokens = 10000
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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_23"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_24 (InputLayer)       [(None, None)]            0
 tf.one_hot_6 (TFOpLambda)   (None, None, 10000)       0

 bidirectional_23 (Bidirect  (None, 64)                2568448
 ional)

 dropout_23 (Dropout)        (None, 64)                0

 dense_23 (Dense)            (None, 1)                 65

=================================================================
Total params: 2568513 (9.80 MB)
Trainable params: 2568513 (9.80 MB)
Non-trainable params: 0 (0.00 Byte)
-----------------------------------------------------------------
```

Training a first basic sequence model

[ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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
50/50 [==============================] - 13s 200ms/step - loss: 0.9255 -
accuracy: 0.5238 - val_loss: 0.6659 - val_accuracy: 0.6430
Epoch 2/10
50/50 [==============================] - 9s 185ms/step - loss: 0.6427 -
accuracy: 0.6237 - val_loss: 0.6231 - val_accuracy: 0.6980
Epoch 3/10
50/50 [==============================] - 9s 186ms/step - loss: 0.5569 -
accuracy: 0.7437 - val_loss: 0.5746 - val_accuracy: 0.7151
Epoch 4/10
50/50 [==============================] - 9s 183ms/step - loss: 0.4761 -
accuracy: 0.8444 - val_loss: 0.7267 - val_accuracy: 0.7451
Epoch 5/10
50/50 [==============================] - 9s 185ms/step - loss: 0.3946 -
accuracy: 0.8994 - val_loss: 0.7295 - val_accuracy: 0.7602
Epoch 6/10
50/50 [==============================] - 9s 183ms/step - loss: 0.1670 -
accuracy: 0.9494 - val_loss: 0.9272 - val_accuracy: 0.7412
Epoch 7/10
50/50 [==============================] - 9s 182ms/step - loss: 4.8936 -
```

```
accuracy: 0.6338 - val_loss: 7.4211 - val_accuracy: 0.5182
Epoch 8/10
50/50 [==============================] - 9s 183ms/step - loss: 0.9293 -
accuracy: 0.9156 - val_loss: 1.0428 - val_accuracy: 0.7661
Epoch 9/10
50/50 [==============================] - 9s 182ms/step - loss: 0.0999 -
accuracy: 0.9875 - val_loss: 1.9301 - val_accuracy: 0.7793
Epoch 10/10
50/50 [==============================] - 9s 185ms/step - loss: 0.0665 -
accuracy: 0.9856 - val_loss: 1.7318 - val_accuracy: 0.7695
782/782 [==============================] - 12s 13ms/step - loss: 0.5790 -
accuracy: 0.7080
Test acc: 0.708
```

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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_24"

-----------------------------------------------------------------
 Layer (type)                Output Shape              Param #
=================================================================
 input_25 (InputLayer)       [(None, None)]            0

 embedding_14 (Embedding)    (None, None, 256)         2560000

 bidirectional_24 (Bidirect  (None, 64)                73984
 ional)

 dropout_24 (Dropout)        (None, 64)                0

 dense_24 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)

-----------------------------------------------------------------
```

[ ]:

```python
callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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}")
```

```
Epoch 1/10
50/50 [==============================] - 12s 174ms/step - loss: 0.8631 -
accuracy: 0.5188 - val_loss: 0.6805 - val_accuracy: 0.5563
Epoch 2/10
50/50 [==============================] - 7s 137ms/step - loss: 0.6208 -
accuracy: 0.6525 - val_loss: 0.6161 - val_accuracy: 0.6616
Epoch 3/10
50/50 [==============================] - 6s 126ms/step - loss: 0.5366 -
accuracy: 0.7969 - val_loss: 0.8890 - val_accuracy: 0.6504
Epoch 4/10
50/50 [==============================] - 6s 121ms/step - loss: 0.3111 -
accuracy: 0.8956 - val_loss: 1.0224 - val_accuracy: 0.7226
Epoch 5/10
50/50 [==============================] - 6s 122ms/step - loss: 0.1944 -
accuracy: 0.9556 - val_loss: 1.6019 - val_accuracy: 0.6681
Epoch 6/10
50/50 [==============================] - 6s 120ms/step - loss: 0.3397 -
accuracy: 0.9538 - val_loss: 4.2780 - val_accuracy: 0.6392
Epoch 7/10
50/50 [==============================] - 6s 117ms/step - loss: 0.1624 -
accuracy: 0.9769 - val_loss: 2.2742 - val_accuracy: 0.7195
Epoch 8/10
50/50 [==============================] - 6s 121ms/step - loss: 0.0498 -
accuracy: 0.9931 - val_loss: 1.9188 - val_accuracy: 0.7068
Epoch 9/10
50/50 [==============================] - 5s 105ms/step - loss: 0.0576 -
accuracy: 0.9925 - val_loss: 3.1249 - val_accuracy: 0.6730
Epoch 10/10
50/50 [==============================] - 5s 103ms/step - loss: 0.0747 -
accuracy: 0.9894 - val_loss: 2.4666 - val_accuracy: 0.7003
782/782 [==============================] - 7s 7ms/step - loss: 0.6203 -
accuracy: 0.6575
Test acc: 0.658
```

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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_25"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_26 (InputLayer)       [(None, None)]            0

 embedding_15 (Embedding)    (None, None, 256)         2560000

 bidirectional_25 (Bidirect  (None, 64)                73984
 ional)

 dropout_25 (Dropout)        (None, 64)                0

 dense_25 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```
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}")
```

```
Epoch 1/10
50/50 [==============================] - 19s 212ms/step - loss: 0.8332 -
accuracy: 0.5387 - val_loss: 0.6604 - val_accuracy: 0.5720
Epoch 2/10
50/50 [==============================] - 8s 152ms/step - loss: 0.5950 -
```

```
accuracy: 0.7506 - val_loss: 0.7695 - val_accuracy: 0.6555
Epoch 3/10
50/50 [==============================] - 7s 141ms/step - loss: 0.2887 -
accuracy: 0.9175 - val_loss: 0.8889 - val_accuracy: 0.7430
Epoch 4/10
50/50 [==============================] - 7s 132ms/step - loss: 0.1389 -
accuracy: 0.9644 - val_loss: 1.3124 - val_accuracy: 0.7384
Epoch 5/10
50/50 [==============================] - 6s 127ms/step - loss: 0.0664 -
accuracy: 0.9794 - val_loss: 1.5764 - val_accuracy: 0.7375
Epoch 6/10
50/50 [==============================] - 6s 127ms/step - loss: 0.0618 -
accuracy: 0.9894 - val_loss: 1.9205 - val_accuracy: 0.7311
Epoch 7/10
50/50 [==============================] - 6s 119ms/step - loss: 0.0420 -
accuracy: 0.9956 - val_loss: 2.7730 - val_accuracy: 0.7381
Epoch 8/10
50/50 [==============================] - 6s 120ms/step - loss: 0.0630 -
accuracy: 0.9900 - val_loss: 2.2939 - val_accuracy: 0.7185
Epoch 9/10
50/50 [==============================] - 6s 117ms/step - loss: 0.0373 -
accuracy: 0.9950 - val_loss: 2.2609 - val_accuracy: 0.7380
Epoch 10/10
50/50 [==============================] - 6s 114ms/step - loss: 0.0381 -
accuracy: 0.9944 - val_loss: 2.3825 - val_accuracy: 0.7249
782/782 [==============================] - 10s 8ms/step - loss: 0.6610 -
accuracy: 0.5739
Test acc: 0.574
```

[ ]:

```
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="relu")(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,
                                    monitor="val_loss")
]
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_26"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_27 (InputLayer)       [(None, None)]            0

 embedding_10 (Embedding)    (None, None, 256)         2560000

 bidirectional_26 (Bidirect  (None, 64)                73984
 ional)

 dropout_26 (Dropout)        (None, 64)                0

 dense_26 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
50/50 [==============================] - 12s 188ms/step - loss: 0.8277 -
accuracy: 0.5194 - val_loss: 0.6730 - val_accuracy: 0.5811
Epoch 2/10
50/50 [==============================] - 7s 146ms/step - loss: 0.6300 -
accuracy: 0.6500 - val_loss: 0.6192 - val_accuracy: 0.6612
Epoch 3/10
50/50 [==============================] - 6s 129ms/step - loss: 0.4826 -
accuracy: 0.8006 - val_loss: 0.8688 - val_accuracy: 0.6552
Epoch 4/10
50/50 [==============================] - 6s 124ms/step - loss: 0.3695 -
accuracy: 0.9056 - val_loss: 0.9261 - val_accuracy: 0.7143
Epoch 5/10
50/50 [==============================] - 6s 111ms/step - loss: 0.2586 -
accuracy: 0.9300 - val_loss: 1.2047 - val_accuracy: 0.7021
Epoch 6/10
50/50 [==============================] - 6s 119ms/step - loss: 0.1648 -
accuracy: 0.9731 - val_loss: 1.6635 - val_accuracy: 0.7360
Epoch 7/10
50/50 [==============================] - 6s 113ms/step - loss: 0.1256 -
accuracy: 0.9762 - val_loss: 2.4898 - val_accuracy: 0.6707
Epoch 8/10
50/50 [==============================] - 5s 107ms/step - loss: 0.0627 -
accuracy: 0.9906 - val_loss: 2.1906 - val_accuracy: 0.7283
```

```
Epoch 9/10
50/50 [==============================] - 5s 100ms/step - loss: 0.0418 -
accuracy: 0.9950 - val_loss: 2.4632 - val_accuracy: 0.7147
Epoch 10/10
50/50 [==============================] - 5s 109ms/step - loss: 0.0553 -
accuracy: 0.9906 - val_loss: 2.8267 - val_accuracy: 0.7294
782/782 [==============================] - 7s 7ms/step - loss: 0.6186 -
accuracy: 0.6593
Test acc: 0.659
```

3 Training Sample Size: 1600

[ ]:
```
!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  17.4M      0  0:00:04  0:00:04 --:--:-- 18.8M
```

[ ]:
```
shutil.rmtree('aclImdb/val')
```

[ ]:
```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_n = base_dir / "train_n"
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 = 10000
    val_files = files[-num_val_samples:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)
```

[ ]:
```
shutil.rmtree('aclImdb/train_n')
```

[ ]:
```
for category in ("neg", "pos"):
    os.makedirs(train_n / category)
    files = os.listdir(train_dir / category)
    num_train_samples=1600
    train_files = files[-num_train_samples:]
    for fname in train_files:
        shutil.move(train_dir / category / fname,
                    train_n / category / fname)

train_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/train_n", 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 3200 files belonging to 2 classes.
Found 20000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

[ ]:

```
from tensorflow.keras import layers

max_length = 150
max_tokens = 10000
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="relu")(x)
```

```
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_27"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_28 (InputLayer)       [(None, None)]            0

 tf.one_hot_7 (TFOpLambda)   (None, None, 10000)       0

 bidirectional_27 (Bidirect  (None, 64)                2568448
 ional)

 dropout_27 (Dropout)        (None, 64)                0

 dense_27 (Dense)            (None, 1)                 65

=================================================================
Total params: 2568513 (9.80 MB)
Trainable params: 2568513 (9.80 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training a first basic sequence model

[ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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
100/100 [==============================] - 14s 110ms/step - loss: 0.8218 -
accuracy: 0.5416 - val_loss: 0.6312 - val_accuracy: 0.7121
Epoch 2/10
100/100 [==============================] - 10s 104ms/step - loss: 0.6078 -
accuracy: 0.7041 - val_loss: 0.5103 - val_accuracy: 0.7542
Epoch 3/10
100/100 [==============================] - 10s 102ms/step - loss: 0.5729 -
accuracy: 0.8328 - val_loss: 0.7749 - val_accuracy: 0.7243
```

```
Epoch 4/10
100/100 [==============================] - 10s 102ms/step - loss: 0.3805 -
accuracy: 0.8856 - val_loss: 0.6372 - val_accuracy: 0.7837
Epoch 5/10
100/100 [==============================] - 10s 102ms/step - loss: 0.2338 -
accuracy: 0.9422 - val_loss: 0.9340 - val_accuracy: 0.7964
Epoch 6/10
100/100 [==============================] - 10s 102ms/step - loss: 0.1880 -
accuracy: 0.9444 - val_loss: 1.2775 - val_accuracy: 0.7919
Epoch 7/10
100/100 [==============================] - 10s 102ms/step - loss: 0.2000 -
accuracy: 0.9684 - val_loss: 1.3876 - val_accuracy: 0.7513
Epoch 8/10
100/100 [==============================] - 10s 102ms/step - loss: 0.1701 -
accuracy: 0.9716 - val_loss: 1.8784 - val_accuracy: 0.7976
Epoch 9/10
100/100 [==============================] - 10s 102ms/step - loss: 0.1597 -
accuracy: 0.9722 - val_loss: 1.2636 - val_accuracy: 0.7676
Epoch 10/10
100/100 [==============================] - 10s 101ms/step - loss: 0.0732 -
accuracy: 0.9872 - val_loss: 1.4698 - val_accuracy: 0.8084
782/782 [==============================] - 12s 13ms/step - loss: 0.5138 -
accuracy: 0.7413
Test acc: 0.741
```

Model that uses an Embedding layer trained from scratch

[ ]:

```python
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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_28"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_29 (InputLayer)       [(None, None)]            0

 embedding_16 (Embedding)    (None, None, 256)         2560000

 bidirectional_28 (Bidirect  (None, 64)                73984
 ional)
 dropout_28 (Dropout)        (None, 64)                0

 dense_28 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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}")
```

```
Epoch 1/10
100/100 [==============================] - 14s 114ms/step - loss: 0.8118 -
accuracy: 0.5638 - val_loss: 0.5774 - val_accuracy: 0.7001
Epoch 2/10
100/100 [==============================] - 9s 85ms/step - loss: 0.6155 -
accuracy: 0.7534 - val_loss: 0.5837 - val_accuracy: 0.7446
Epoch 3/10
100/100 [==============================] - 7s 73ms/step - loss: 0.4329 -
accuracy: 0.8637 - val_loss: 1.2133 - val_accuracy: 0.7515
Epoch 4/10
100/100 [==============================] - 7s 71ms/step - loss: 0.3596 -
accuracy: 0.9109 - val_loss: 1.0515 - val_accuracy: 0.7661
Epoch 5/10
100/100 [==============================] - 6s 62ms/step - loss: 0.2342 -
accuracy: 0.9472 - val_loss: 1.4572 - val_accuracy: 0.7738
Epoch 6/10
100/100 [==============================] - 6s 61ms/step - loss: 0.1907 -
accuracy: 0.9681 - val_loss: 1.7063 - val_accuracy: 0.7720
Epoch 7/10
100/100 [==============================] - 6s 60ms/step - loss: 0.1384 -
accuracy: 0.9797 - val_loss: 2.0417 - val_accuracy: 0.7863
Epoch 8/10
100/100 [==============================] - 6s 60ms/step - loss: 0.1291 -
accuracy: 0.9834 - val_loss: 2.0746 - val_accuracy: 0.7487
Epoch 9/10
100/100 [==============================] - 6s 58ms/step - loss: 0.1194 -
accuracy: 0.9850 - val_loss: 2.3379 - val_accuracy: 0.7764
```

```
Epoch 10/10
100/100 [==============================] - 6s 57ms/step - loss: 0.1249 -
accuracy: 0.9859 - val_loss: 2.2613 - val_accuracy: 0.7789
782/782 [==============================] - 7s 7ms/step - loss: 0.5812 -
accuracy: 0.6968
Test acc: 0.697
```

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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_29"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_30 (InputLayer)       [(None, None)]            0

 embedding_17 (Embedding)    (None, None, 256)         2560000

 bidirectional_29 (Bidirect  (None, 64)                73984
 ional)

 dropout_29 (Dropout)        (None, 64)                0

 dense_29 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```
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}")
```

```
Epoch 1/10
100/100 [==============================] - 20s 133ms/step - loss: 0.7886 -
accuracy: 0.6012 - val_loss: 0.5487 - val_accuracy: 0.7365
Epoch 2/10
100/100 [==============================] - 9s 93ms/step - loss: 0.5285 -
accuracy: 0.8059 - val_loss: 0.7809 - val_accuracy: 0.7398
Epoch 3/10
100/100 [==============================] - 8s 82ms/step - loss: 0.3784 -
accuracy: 0.9041 - val_loss: 1.6285 - val_accuracy: 0.7845
Epoch 4/10
100/100 [==============================] - 8s 79ms/step - loss: 0.2897 -
accuracy: 0.9491 - val_loss: 1.5406 - val_accuracy: 0.7961
Epoch 5/10
100/100 [==============================] - 7s 70ms/step - loss: 0.2180 -
accuracy: 0.9613 - val_loss: 1.4738 - val_accuracy: 0.7815
Epoch 6/10
100/100 [==============================] - 7s 66ms/step - loss: 0.1592 -
accuracy: 0.9775 - val_loss: 1.8206 - val_accuracy: 0.7868
Epoch 7/10
100/100 [==============================] - 7s 67ms/step - loss: 0.1099 -
accuracy: 0.9856 - val_loss: 1.7729 - val_accuracy: 0.7749
Epoch 8/10
100/100 [==============================] - 7s 65ms/step - loss: 0.1018 -
accuracy: 0.9894 - val_loss: 2.0727 - val_accuracy: 0.7806
Epoch 9/10
100/100 [==============================] - 7s 67ms/step - loss: 0.0946 -
accuracy: 0.9909 - val_loss: 2.1224 - val_accuracy: 0.7843
Epoch 10/10
100/100 [==============================] - 7s 65ms/step - loss: 0.0827 -
accuracy: 0.9903 - val_loss: 2.2345 - val_accuracy: 0.7779
782/782 [==============================] - 10s 7ms/step - loss: 0.5541 -
accuracy: 0.7238
Test acc: 0.724
```

[ ]:

```
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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```python
callbacks = [
    keras.callbacks.ModelCheckpoint("glove_embeddings_sequence_model.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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_30"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_31 (InputLayer)       [(None, None)]            0

 embedding_10 (Embedding)    (None, None, 256)         2560000

 bidirectional_30 (Bidirect  (None, 64)                73984
 ional)

 dropout_30 (Dropout)        (None, 64)                0

 dense_30 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
100/100 [==============================] - 14s 112ms/step - loss: 0.8863 -
accuracy: 0.5213 - val_loss: 0.6534 - val_accuracy: 0.6193
Epoch 2/10
100/100 [==============================] - 8s 83ms/step - loss: 0.6214 -
accuracy: 0.6822 - val_loss: 0.5894 - val_accuracy: 0.6883
Epoch 3/10
100/100 [==============================] - 7s 73ms/step - loss: 0.4975 -
accuracy: 0.8178 - val_loss: 1.0953 - val_accuracy: 0.7497
Epoch 4/10
100/100 [==============================] - 8s 75ms/step - loss: 0.4327 -
accuracy: 0.8872 - val_loss: 1.0923 - val_accuracy: 0.7519
Epoch 5/10
100/100 [==============================] - 6s 64ms/step - loss: 0.2546 -
accuracy: 0.9350 - val_loss: 1.4395 - val_accuracy: 0.7624
Epoch 6/10
```

```
100/100 [==============================] - 6s 59ms/step - loss: 0.2205 -
accuracy: 0.9591 - val_loss: 1.8525 - val_accuracy: 0.7645
Epoch 7/10
100/100 [==============================] - 6s 62ms/step - loss: 0.1214 -
accuracy: 0.9806 - val_loss: 1.9674 - val_accuracy: 0.7679
Epoch 8/10
100/100 [==============================] - 6s 62ms/step - loss: 0.1195 -
accuracy: 0.9825 - val_loss: 2.0161 - val_accuracy: 0.7764
Epoch 9/10
100/100 [==============================] - 6s 60ms/step - loss: 0.1043 -
accuracy: 0.9853 - val_loss: 2.2058 - val_accuracy: 0.7404
Epoch 10/10
100/100 [==============================] - 6s 56ms/step - loss: 0.0746 -
accuracy: 0.9881 - val_loss: 2.2890 - val_accuracy: 0.7645
782/782 [==============================] - 7s 7ms/step - loss: 0.5913 -
accuracy: 0.6807
Test acc: 0.681
```

4 Training Sample Size: 2400

[ ]:

```
!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  42.5M      0  0:00:01  0:00:01 --:--:-- 42.5M
```

[ ]:

```
shutil.rmtree('aclImdb/val')
```

[ ]:

```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_n = base_dir / "train_n"
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 = 10000
    val_files = files[-num_val_samples:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)
```

```
[ ]: shutil.rmtree('aclImdb/train_n')
```

```
[ ]: for category in ("neg", "pos"):
         os.makedirs(train_n / category)
         files = os.listdir(train_dir / category)
         num_train_samples=2400
         train_files = files[-num_train_samples:]
         for fname in train_files:
             shutil.move(train_dir / category / fname,
                         train_n / category / fname)

     train_ds = keras.utils.text_dataset_from_directory(
         "aclImdb/train_n", 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 4800 files belonging to 2 classes.
Found 20000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

```
[ ]: from tensorflow.keras import layers

     max_length = 150
     max_tokens = 10000
     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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_36"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_37 (InputLayer)       [(None, None)]            0

 tf.one_hot_10 (TFOpLambda)  (None, None, 10000)       0

 bidirectional_36 (Bidirect  (None, 64)                2568448
 ional)

 dropout_36 (Dropout)        (None, 64)                0

 dense_36 (Dense)            (None, 1)                 65

=================================================================
Total params: 2568513 (9.80 MB)
Trainable params: 2568513 (9.80 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training a first basic sequence model

[ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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
150/150 [==============================] - 15s 80ms/step - loss: 0.7865 -
accuracy: 0.5508 - val_loss: 0.6253 - val_accuracy: 0.6247
```

```
Epoch 2/10
150/150 [==============================] - 11s 76ms/step - loss: 0.6022 -
accuracy: 0.7202 - val_loss: 0.5981 - val_accuracy: 0.6761
Epoch 3/10
150/150 [==============================] - 11s 75ms/step - loss: 0.5621 -
accuracy: 0.8325 - val_loss: 0.5605 - val_accuracy: 0.7801
Epoch 4/10
150/150 [==============================] - 11s 75ms/step - loss: 0.5077 -
accuracy: 0.8392 - val_loss: 4.1745 - val_accuracy: 0.6895
Epoch 5/10
150/150 [==============================] - 11s 74ms/step - loss: 0.4082 -
accuracy: 0.8867 - val_loss: 0.6043 - val_accuracy: 0.7845
Epoch 6/10
150/150 [==============================] - 11s 75ms/step - loss: 0.3092 -
accuracy: 0.9148 - val_loss: 0.8699 - val_accuracy: 0.7756
Epoch 7/10
150/150 [==============================] - 11s 74ms/step - loss: 0.2727 -
accuracy: 0.9456 - val_loss: 1.2370 - val_accuracy: 0.8005
Epoch 8/10
150/150 [==============================] - 11s 75ms/step - loss: 0.2693 -
accuracy: 0.9575 - val_loss: 1.4375 - val_accuracy: 0.8057
Epoch 9/10
150/150 [==============================] - 11s 74ms/step - loss: 0.1609 -
accuracy: 0.9694 - val_loss: 1.7880 - val_accuracy: 0.7965
Epoch 10/10
150/150 [==============================] - 11s 75ms/step - loss: 0.1742 -
accuracy: 0.9754 - val_loss: 1.6301 - val_accuracy: 0.8143
782/782 [==============================] - 12s 13ms/step - loss: 0.5762 -
accuracy: 0.7639
Test acc: 0.764
```

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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_32"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_33 (InputLayer)       [(None, None)]            0

 embedding_18 (Embedding)    (None, None, 256)         2560000

 bidirectional_32 (Bidirect  (None, 64)                73984
 ional)

 dropout_32 (Dropout)        (None, 64)                0

 dense_32 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru.keras",
                                    save_best_only=True,
                                    monitor="val_loss")
]
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}")
```

```
Epoch 1/10
150/150 [==============================] - 16s 88ms/step - loss: 0.7235 -
accuracy: 0.5821 - val_loss: 0.8316 - val_accuracy: 0.6507
Epoch 2/10
150/150 [==============================] - 9s 62ms/step - loss: 0.5940 -
accuracy: 0.7902 - val_loss: 0.6608 - val_accuracy: 0.7768
Epoch 3/10
150/150 [==============================] - 8s 56ms/step - loss: 0.4604 -
accuracy: 0.8683 - val_loss: 1.4596 - val_accuracy: 0.7707
Epoch 4/10
150/150 [==============================] - 7s 48ms/step - loss: 0.3750 -
accuracy: 0.9165 - val_loss: 3.4881 - val_accuracy: 0.6936
Epoch 5/10
150/150 [==============================] - 7s 45ms/step - loss: 0.2657 -
accuracy: 0.9463 - val_loss: 1.3386 - val_accuracy: 0.8046
Epoch 6/10
150/150 [==============================] - 7s 46ms/step - loss: 0.2947 -
accuracy: 0.9592 - val_loss: 1.6646 - val_accuracy: 0.7803
Epoch 7/10
150/150 [==============================] - 7s 45ms/step - loss: 0.1998 -
accuracy: 0.9750 - val_loss: 2.5277 - val_accuracy: 0.7686
```

```
Epoch 8/10
150/150 [==============================] - 7s 45ms/step - loss: 0.1695 -
accuracy: 0.9775 - val_loss: 2.0419 - val_accuracy: 0.8027
Epoch 9/10
150/150 [==============================] - 7s 45ms/step - loss: 0.1595 -
accuracy: 0.9798 - val_loss: 2.7800 - val_accuracy: 0.7483
Epoch 10/10
150/150 [==============================] - 7s 44ms/step - loss: 0.1356 -
accuracy: 0.9848 - val_loss: 2.1583 - val_accuracy: 0.7964
782/782 [==============================] - 7s 7ms/step - loss: 0.7168 -
accuracy: 0.7593
Test acc: 0.759
```

Using an `Embedding` layer with masking enabled

[ ]:

```python
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="relu")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Model: "model_33"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_34 (InputLayer)       [(None, None)]            0

 embedding_19 (Embedding)    (None, None, 256)         2560000

 bidirectional_33 (Bidirect  (None, 64)                73984
 ional)

 dropout_33 (Dropout)        (None, 64)                0

 dense_33 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```
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}")
```

```
Epoch 1/10
150/150 [==============================] - 24s 105ms/step - loss: 0.7981 -
accuracy: 0.5810 - val_loss: 0.6535 - val_accuracy: 0.6593
Epoch 2/10
150/150 [==============================] - 11s 72ms/step - loss: 0.5778 -
accuracy: 0.7715 - val_loss: 0.7717 - val_accuracy: 0.7883
Epoch 3/10
150/150 [==============================] - 9s 60ms/step - loss: 0.4565 -
accuracy: 0.8481 - val_loss: 0.6678 - val_accuracy: 0.7617
Epoch 4/10
150/150 [==============================] - 8s 56ms/step - loss: 0.3105 -
accuracy: 0.9112 - val_loss: 1.1266 - val_accuracy: 0.7893
Epoch 5/10
150/150 [==============================] - 8s 51ms/step - loss: 0.2642 -
accuracy: 0.9450 - val_loss: 1.6530 - val_accuracy: 0.7831
Epoch 6/10
150/150 [==============================] - 7s 50ms/step - loss: 0.1855 -
accuracy: 0.9669 - val_loss: 1.8064 - val_accuracy: 0.7765
Epoch 7/10
150/150 [==============================] - 8s 50ms/step - loss: 0.1610 -
accuracy: 0.9748 - val_loss: 1.9281 - val_accuracy: 0.7866
Epoch 8/10
150/150 [==============================] - 7s 50ms/step - loss: 0.1386 -
accuracy: 0.9810 - val_loss: 1.9349 - val_accuracy: 0.7974
Epoch 9/10
150/150 [==============================] - 8s 51ms/step - loss: 0.1206 -
accuracy: 0.9842 - val_loss: 2.1599 - val_accuracy: 0.7972
Epoch 10/10
150/150 [==============================] - 7s 49ms/step - loss: 0.0975 -
accuracy: 0.9885 - val_loss: 2.1211 - val_accuracy: 0.7750
782/782 [==============================] - 10s 7ms/step - loss: 0.6617 -
accuracy: 0.6500
Test acc: 0.650
```

[ ]:

```
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = embedding_layer(inputs)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
```

```python
outputs = layers.Dense(1, activation="relu")(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,
                                    monitor="val_loss")
]
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_38"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_39 (InputLayer)       [(None, None)]            0

 embedding_10 (Embedding)    (None, None, 256)         2560000

 bidirectional_38 (Bidirect  (None, 64)                73984
 ional)

 dropout_38 (Dropout)        (None, 64)                0

 dense_38 (Dense)            (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
150/150 [==============================] - 16s 90ms/step - loss: 0.4112 -
accuracy: 0.8756 - val_loss: 1.3057 - val_accuracy: 0.7925
Epoch 2/10
150/150 [==============================] - 10s 65ms/step - loss: 0.1214 -
accuracy: 0.9817 - val_loss: 1.9420 - val_accuracy: 0.8055
Epoch 3/10
150/150 [==============================] - 7s 50ms/step - loss: 0.0813 -
accuracy: 0.9896 - val_loss: 1.9894 - val_accuracy: 0.8033
Epoch 4/10
```

```
150/150 [==============================] - 7s 49ms/step - loss: 0.0614 -
accuracy: 0.9917 - val_loss: 2.2024 - val_accuracy: 0.7986
Epoch 5/10
150/150 [==============================] - 8s 50ms/step - loss: 0.0575 -
accuracy: 0.9915 - val_loss: 2.3071 - val_accuracy: 0.7968
Epoch 6/10
150/150 [==============================] - 7s 44ms/step - loss: 0.0603 -
accuracy: 0.9923 - val_loss: 2.2915 - val_accuracy: 0.7942
Epoch 7/10
150/150 [==============================] - 6s 42ms/step - loss: 0.0604 -
accuracy: 0.9931 - val_loss: 2.4497 - val_accuracy: 0.7890
Epoch 8/10
150/150 [==============================] - 7s 44ms/step - loss: 0.0732 -
accuracy: 0.9923 - val_loss: 2.3744 - val_accuracy: 0.8005
Epoch 9/10
150/150 [==============================] - 7s 44ms/step - loss: 0.0578 -
accuracy: 0.9942 - val_loss: 2.4252 - val_accuracy: 0.8022
Epoch 10/10
150/150 [==============================] - 6s 42ms/step - loss: 0.0839 -
accuracy: 0.9917 - val_loss: 2.3643 - val_accuracy: 0.7939
782/782 [==============================] - 7s 7ms/step - loss: 1.3381 -
accuracy: 0.7795
Test acc: 0.780
```