Assignment4_yli130

November 20, 2023

0.1 Summary for Assignment_4

Yanxi Li, yli130@kent.edu

For each model test accuracy are as follows:
1. Model in the reference book with 600 review words: 0.886
2. Question_1 cutoff reviews after 150 words: 0.846
3. Question_2 retrict training sample to 160: 0.534
4. Question_3 validate on 10,000 samples: 0.864
5. Question_4 only top 10,000 tokens: 0.877
6. Question_5_1 20,000 training with embedding layer: 0.874
7. Question_5_2 20,000 training with pretrained word embedding: 0.876
8. Question_5_3 15,000 training with embedding layer: 0.867
9. Question_5_4 22,500 training with embedding layer: 0.88
10. Question_5_5 160 training with pretrained word embedding: 0.573
11. Question_5_6 160 training with embedding layer: 0.655
12. Question_5_7 2,2500 training with pretrained word embedding:0.865

From Question_1 to Question_4 with one-hot encoding, more reviews, more tokens, more training samples could reach higher test accuracy for the model.

The review words and tokens used in Question_5 is 600, 20,000, respectively.

When the training data is 20,000, using the embedding layer and pretrained word embedding has the similar result, around 0.87.

As for small training dataset, like 160 training samples, although embedding test accuracy is higher than pretrained word embedding, embedding layer shows overfit soon. The training accuracy almost reach to 100% while validation accuracy only around 50%.

When the training data size is larger, for example, 2,2500, the test accuracy for embedding layer is higher than pretrained word embedding. Since the dataset has enough samples to learn, leveraging pretrained embeddings is not very helpful in this case. But for small training samples in the above illustration, pretrained embeddings worked.

0.2 Model in the book

0.2.1 Download the data

[1]:

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

0.2.2 Preparing the data

[2]:

```
'''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)'''
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
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.
```

0.2.3 Preparing integer sequence datasets

[3]:

```python
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)
```

0.2.4 A sequence model built on one-hot encoded vector sequences

[4]:

```python
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 (Bidirectional (None, 64)                5128448

_________________________________________________________________
```

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

_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 5,128,513
Trainable params: 5,128,513
Non-trainable params: 0
_________________________________________________________________
```

0.2.5 Train and test the model

[5]:
```
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 [==============================] - 76s 116ms/step - loss: 0.5203 -
accuracy: 0.7549 - val_loss: 0.3529 - val_accuracy: 0.8786
Epoch 2/10
625/625 [==============================] - 72s 115ms/step - loss: 0.3360 -
accuracy: 0.8772 - val_loss: 0.3107 - val_accuracy: 0.8756
Epoch 3/10
625/625 [==============================] - 72s 115ms/step - loss: 0.2738 -
accuracy: 0.9032 - val_loss: 0.2798 - val_accuracy: 0.8922
Epoch 4/10
625/625 [==============================] - 72s 115ms/step - loss: 0.2343 -
accuracy: 0.9161 - val_loss: 0.6369 - val_accuracy: 0.7568
Epoch 5/10
625/625 [==============================] - 72s 115ms/step - loss: 0.2124 -
accuracy: 0.9277 - val_loss: 0.3253 - val_accuracy: 0.8788
Epoch 6/10
625/625 [==============================] - 72s 115ms/step - loss: 0.1820 -
accuracy: 0.9379 - val_loss: 0.3287 - val_accuracy: 0.8904
Epoch 7/10
625/625 [==============================] - 72s 115ms/step - loss: 0.1634 -
accuracy: 0.9442 - val_loss: 0.3241 - val_accuracy: 0.8696
Epoch 8/10
625/625 [==============================] - 72s 115ms/step - loss: 0.1491 -
accuracy: 0.9514 - val_loss: 0.3696 - val_accuracy: 0.8784
Epoch 9/10
625/625 [==============================] - 72s 115ms/step - loss: 0.1337 -
accuracy: 0.9561 - val_loss: 0.3962 - val_accuracy: 0.8832
Epoch 10/10
```

```
625/625 [==============================] - 72s 115ms/step - loss: 0.1142 -
accuracy: 0.9632 - val_loss: 0.3573 - val_accuracy: 0.8558
782/782 [==============================] - 42s 52ms/step - loss: 0.2969 -
accuracy: 0.8859
Test acc: 0.886
```

Question_1

November 20, 2023

0.1 Question_1 Cutoff reviews after 150 words

0.1.1 Preparing the data

[1]:

```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
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.
```

0.1.2 Preparing integer sequence datasets

[2]:

```
from tensorflow.keras import layers

max_length = 150
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)
```

0.1.3 A sequence model built on one-hot encoded vector sequences

[3]:

```
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 (Bidirectional (None, 64)                5128448
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 5,128,513
Trainable params: 5,128,513
Non-trainable params: 0
_________________________________________________________________
```

0.1.4 Train and test the model

[4]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm_150words.keras",
                                    save_best_only=True)
]
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10,␣
 ↪callbacks=callbacks)
```

```python
model = keras.models.load_model("one_hot_bidir_lstm_150words.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Epoch 1/10
625/625 [==============================] - 24s 32ms/step - loss: 0.5071 -
accuracy: 0.7621 - val_loss: 0.3978 - val_accuracy: 0.8272
Epoch 2/10
625/625 [==============================] - 20s 31ms/step - loss: 0.3450 -
accuracy: 0.8664 - val_loss: 0.3497 - val_accuracy: 0.8576
Epoch 3/10
625/625 [==============================] - 20s 31ms/step - loss: 0.2737 -
accuracy: 0.8962 - val_loss: 0.3489 - val_accuracy: 0.8652
Epoch 4/10
625/625 [==============================] - 20s 31ms/step - loss: 0.2260 -
accuracy: 0.9168 - val_loss: 0.3793 - val_accuracy: 0.8592
Epoch 5/10
625/625 [==============================] - 20s 31ms/step - loss: 0.1872 -
accuracy: 0.9338 - val_loss: 0.3848 - val_accuracy: 0.8578
Epoch 6/10
625/625 [==============================] - 20s 31ms/step - loss: 0.1536 -
accuracy: 0.9465 - val_loss: 0.4357 - val_accuracy: 0.8556
Epoch 7/10
625/625 [==============================] - 20s 31ms/step - loss: 0.1180 -
accuracy: 0.9596 - val_loss: 0.4509 - val_accuracy: 0.8110
Epoch 8/10
625/625 [==============================] - 20s 31ms/step - loss: 0.0911 -
accuracy: 0.9705 - val_loss: 0.4429 - val_accuracy: 0.8320
Epoch 9/10
625/625 [==============================] - 20s 31ms/step - loss: 0.0680 -
accuracy: 0.9781 - val_loss: 0.5979 - val_accuracy: 0.8522
Epoch 10/10
625/625 [==============================] - 20s 31ms/step - loss: 0.0551 -
accuracy: 0.9815 - val_loss: 0.6117 - val_accuracy: 0.8420
782/782 [==============================] - 12s 14ms/step - loss: 0.3837 -
accuracy: 0.8462
Test acc: 0.846
```

Question_2

November 20, 2023

0.1 Question_2 Restrict training sample to 160

0.1.1 Preparing the data

Remove the validation back to train

[1]:
```
'''import os, shutil

val_dir_pos = 'aclImdb/val/pos'
train_dir_pos = 'aclImdb/train/pos'

for file in os.listdir(val_dir_pos):
    val = val_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(val, train)

val_dir_neg = 'aclImdb/val/neg'
train_dir_neg = 'aclImdb/train/neg'

for file in os.listdir(val_dir_neg):
    val = val_dir_neg + '/' + file
    train = train_dir_neg + '/' + file
    shutil.move(val, train)'''
```

[1]:
```
"import os, shutil\n\nval_dir_pos = 'aclImdb/val/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file in os.listdir(val_dir_pos):\n    val =
val_dir_pos +'/'+ file\n    train = train_dir_pos + '/' + file\n
shutil.move(val, train)\n    \nval_dir_neg = 'aclImdb/val/neg'\ntrain_dir_neg =
'aclImdb/train/neg'\n\nfor file in os.listdir(val_dir_neg):\n    val =
val_dir_neg + '/' + file\n    train = train_dir_neg + '/' + file\n
shutil.move(val, train)"
```

Make a small training sample

[2]:
```
'''train_q2_dir_pos = 'aclImdb/train_q2/pos'
train_dir_pos = 'aclImdb/train/pos'

for file_name in (os.listdir(train_dir_pos))[500:600]:
    train = train_dir_pos + '/' + file_name
    small_train = train_q2_dir_pos + '/' + file_name
    shutil.move(train, small_train)

train_q2_dir_neg = 'aclImdb/train_q2/neg'
train_dir_neg = 'aclImdb/train/neg'

for file_name in (os.listdir(train_dir_neg))[500:600]:
    train = train_dir_neg + '/' + file_name
    small_train = train_q2_dir_neg + '/' + file_name
    shutil.move(train, small_train)'''
```

[2]:

```
"train_q2_dir_pos = 'aclImdb/train_q2/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file_name in (os.listdir(train_dir_pos))[500:600]:\n
train = train_dir_pos + '/' + file_name\n    small_train = train_q2_dir_pos +
'/' + file_name\n    shutil.move(train, small_train)\n    \ntrain_q2_dir_neg =
'aclImdb/train_q2/neg'\ntrain_dir_neg = 'aclImdb/train/neg'\n\nfor file_name in
(os.listdir(train_dir_neg))[500:600]:\n    train = train_dir_neg + '/' +
file_name\n    small_train = train_q2_dir_neg + '/' + file_name\n
shutil.move(train, small_train)"
```

Show train, validation, test sample number

[3]:

```python
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_q2"
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_q2", batch_size=batch_size
)   # change the train data directory
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 160 files belonging to 2 classes.
Found 40 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

0.1.2 Preparing integer sequence datasets

[4]:

```
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)
```

0.1.3 A sequence model built on one-hot encoded vector sequences

[5]:

```
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 (Bidirectional (None, 64)                5128448
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 5,128,513
Trainable params: 5,128,513
Non-trainable params: 0
_________________________________________________________________
```

0.1.4 Train and test the model

[6]:

```python
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm_smalltrain.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_smalltrain.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Epoch 1/10
5/5 [==============================] - 5s 296ms/step - loss: 0.6950 - accuracy:
0.4500 - val_loss: 0.6929 - val_accuracy: 0.5500
Epoch 2/10
5/5 [==============================] - 1s 143ms/step - loss: 0.6901 - accuracy:
0.5688 - val_loss: 0.6928 - val_accuracy: 0.5500
Epoch 3/10
5/5 [==============================] - 1s 146ms/step - loss: 0.6848 - accuracy:
0.6562 - val_loss: 0.6925 - val_accuracy: 0.6000
Epoch 4/10
5/5 [==============================] - 1s 144ms/step - loss: 0.6732 - accuracy:
0.8500 - val_loss: 0.6897 - val_accuracy: 0.6000
Epoch 5/10
5/5 [==============================] - 1s 138ms/step - loss: 0.7405 - accuracy:
0.7875 - val_loss: 0.9221 - val_accuracy: 0.5000
Epoch 6/10
5/5 [==============================] - 1s 140ms/step - loss: 0.7738 - accuracy:
0.6187 - val_loss: 0.6815 - val_accuracy: 0.5500
Epoch 7/10
5/5 [==============================] - 1s 150ms/step - loss: 0.6065 - accuracy:
0.8125 - val_loss: 0.6784 - val_accuracy: 0.5750
Epoch 8/10
```

```
5/5 [==============================] - 1s 144ms/step - loss: 0.5479 - accuracy:
0.8813 - val_loss: 0.6739 - val_accuracy: 0.5750
Epoch 9/10
5/5 [==============================] - 1s 138ms/step - loss: 0.5641 - accuracy:
0.7875 - val_loss: 0.6730 - val_accuracy: 0.5750
Epoch 10/10
5/5 [==============================] - 1s 143ms/step - loss: 0.4876 - accuracy:
0.8938 - val_loss: 0.6995 - val_accuracy: 0.5000
782/782 [==============================] - 41s 52ms/step - loss: 0.6798 -
accuracy: 0.5338
Test acc: 0.534
```

Question_3

November 20, 2023

0.1 Question_3 Validate on 10,000 samples

0.1.1 Reorganize the dataset

[1]:

```python
'''import os, shutil

val_dir_pos = 'aclImdb/val/pos'
train_dir_pos = 'aclImdb/train_q2/pos'

for file in os.listdir(val_dir_pos):
    val = val_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(val, train)

val_dir_neg = 'aclImdb/val/neg'
train_dir_neg = 'aclImdb/train_q2/neg'

for file in os.listdir(val_dir_neg):
    val = val_dir_neg + '/' + file
    train = train_dir_neg + '/' + file
    shutil.move(val, train)'''
```

[1]:

```
"import os, shutil\n\nval_dir_pos = 'aclImdb/val/pos'\ntrain_dir_pos =
'aclImdb/train_q2/pos'\n\nfor file in os.listdir(val_dir_pos):\n    val =
val_dir_pos +'/'+ file\n    train = train_dir_pos + '/' + file\n
shutil.move(val, train)\n    \nval_dir_neg = 'aclImdb/val/neg'\ntrain_dir_neg =
'aclImdb/train_q2/neg'\n\nfor file in os.listdir(val_dir_neg):\n    val =
val_dir_neg + '/' + file\n    train = train_dir_neg + '/' + file\n
shutil.move(val, train)"
```

[2]:

```python
'''train_q2_dir_pos = 'aclImdb/train_q2/pos'
train_dir_pos = 'aclImdb/train/pos'

for file in os.listdir(train_q2_dir_pos):
    small_train = train_q2_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(small_train, train)
```

```
train_q2_dir_neg = 'aclImdb/train_q2/neg'
train_dir_neg = 'aclImdb/train/neg'

for file in os.listdir(train_q2_dir_neg):
    small_train = train_q2_dir_neg +'/'+ file
    train = train_dir_neg + '/' + file
    shutil.move(small_train, train)'''
```

[2]:

```
"train_q2_dir_pos = 'aclImdb/train_q2/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file in os.listdir(train_q2_dir_pos):\n
small_train = train_q2_dir_pos +'/'+ file\n    train = train_dir_pos + '/' +
file\n    shutil.move(small_train, train)\n    \ntrain_q2_dir_neg =
'aclImdb/train_q2/neg'\ntrain_dir_neg = 'aclImdb/train/neg'\n\nfor file in
os.listdir(train_q2_dir_neg):\n    small_train = train_q2_dir_neg +'/'+ file\n
train = train_dir_neg + '/' + file\n    shutil.move(small_train, train)"
```

0.1.2 Preparing the data

[3]:

```
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(0.4 * 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
)   # change the train data directory
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 15000 files belonging to 2 classes.
Found 10000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

0.1.3 Preparing integer sequence datasets

[4]:

```python
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)
```

0.1.4 A sequence model built on one-hot encoded vector sequences

[5]:

```python
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 (Bidirectional (None, 64)                5128448
 )

_________________________________________________________________
dropout (Dropout)            (None, 64)                0

_________________________________________________________________
dense (Dense)                (None, 1)                 65

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

_________________________________________________________________
```

0.1.5 Train and test the model

[6]:
```python
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm_vali10000.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_vali10000.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Epoch 1/10
469/469 [==============================] - 67s 136ms/step - loss: 0.5690 -
accuracy: 0.7110 - val_loss: 0.4378 - val_accuracy: 0.8377
Epoch 2/10
469/469 [==============================] - 63s 135ms/step - loss: 0.3656 -
accuracy: 0.8679 - val_loss: 0.3033 - val_accuracy: 0.8797
Epoch 3/10
469/469 [==============================] - 63s 135ms/step - loss: 0.2842 -
accuracy: 0.8983 - val_loss: 0.2830 - val_accuracy: 0.8868
Epoch 4/10
469/469 [==============================] - 64s 135ms/step - loss: 0.2345 -
accuracy: 0.9210 - val_loss: 0.3388 - val_accuracy: 0.8539
Epoch 5/10
469/469 [==============================] - 64s 136ms/step - loss: 0.1939 -
accuracy: 0.9363 - val_loss: 0.3432 - val_accuracy: 0.8602
Epoch 6/10
469/469 [==============================] - 63s 135ms/step - loss: 0.1637 -
accuracy: 0.9448 - val_loss: 0.3528 - val_accuracy: 0.8866
Epoch 7/10
469/469 [==============================] - 64s 135ms/step - loss: 0.1529 -
accuracy: 0.9502 - val_loss: 0.4134 - val_accuracy: 0.8801
Epoch 8/10
469/469 [==============================] - 64s 135ms/step - loss: 0.1215 -
accuracy: 0.9621 - val_loss: 0.4512 - val_accuracy: 0.8798
Epoch 9/10
```

```
469/469 [==============================] - 64s 135ms/step - loss: 0.1068 -
accuracy: 0.9658 - val_loss: 0.4775 - val_accuracy: 0.8651
Epoch 10/10
469/469 [==============================] - 64s 135ms/step - loss: 0.0831 -
accuracy: 0.9743 - val_loss: 0.4113 - val_accuracy: 0.8790
782/782 [==============================] - 41s 52ms/step - loss: 0.3147 -
accuracy: 0.8637
Test acc: 0.864
```

Question_4

November 20, 2023

0.1 Question_4 Only top 10,000 words

0.1.1 Preparing the data

[1]:

```python
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.
```

0.1.2 Preparing integer sequence datasets

[2]:

```
from tensorflow.keras import layers

max_length = 600
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)
```

0.1.3 A sequence model built on one-hot encoded vector sequences

[3]:

```
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, 10000)       0

_________________________________________________________________
bidirectional (Bidirectional (None, 64)                2568448

_________________________________________________________________
```

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

_________________________________________________________________
dense (Dense)                (None, 1)                65
=================================================================
Total params: 2,568,513
Trainable params: 2,568,513
Non-trainable params: 0

_________________________________________________________________
```

0.1.4 Train and test the model

[4]:

```python
callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm_10000words.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_10000words.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Epoch 1/10
625/625 [==============================] - 56s 82ms/step - loss: 0.5290 -
accuracy: 0.7520 - val_loss: 0.4038 - val_accuracy: 0.8456
Epoch 2/10
625/625 [==============================] - 50s 80ms/step - loss: 0.3535 -
accuracy: 0.8680 - val_loss: 0.2894 - val_accuracy: 0.8846
Epoch 3/10
625/625 [==============================] - 50s 80ms/step - loss: 0.2905 -
accuracy: 0.8958 - val_loss: 0.2969 - val_accuracy: 0.8842
Epoch 4/10
625/625 [==============================] - 50s 81ms/step - loss: 0.2535 -
accuracy: 0.9112 - val_loss: 0.2904 - val_accuracy: 0.8850
Epoch 5/10
625/625 [==============================] - 50s 80ms/step - loss: 0.2261 -
accuracy: 0.9222 - val_loss: 0.2905 - val_accuracy: 0.8856
Epoch 6/10
625/625 [==============================] - 50s 81ms/step - loss: 0.2077 -
accuracy: 0.9280 - val_loss: 0.3219 - val_accuracy: 0.8842
Epoch 7/10
625/625 [==============================] - 50s 81ms/step - loss: 0.2033 -
accuracy: 0.9327 - val_loss: 0.3142 - val_accuracy: 0.8846
Epoch 8/10
625/625 [==============================] - 50s 81ms/step - loss: 0.1760 -
accuracy: 0.9413 - val_loss: 0.4033 - val_accuracy: 0.8854
Epoch 9/10
625/625 [==============================] - 50s 81ms/step - loss: 0.1637 -
accuracy: 0.9438 - val_loss: 0.4176 - val_accuracy: 0.8748
Epoch 10/10
```

```
625/625 [==============================] - 51s 81ms/step - loss: 0.1531 -
accuracy: 0.9505 - val_loss: 0.3691 - val_accuracy: 0.8844
782/782 [==============================] - 31s 38ms/step - loss: 0.3027 -
accuracy: 0.8772
Test acc: 0.877
```

Question_5_1

November 20, 2023

0.1 Question_5 20,000 training with embedding layer

0.1.1 Reorganize the dataset

[1]:

```
'''import os, shutil

val_dir_pos = 'aclImdb/val/pos'
train_dir_pos = 'aclImdb/train/pos'

for file in os.listdir(val_dir_pos):
    val = val_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(val, train)

val_dir_neg = 'aclImdb/val/neg'
train_dir_neg = 'aclImdb/train/neg'

for file in os.listdir(val_dir_neg):
    val = val_dir_neg + '/' + file
    train = train_dir_neg + '/' + file
    shutil.move(val, train)'''
```

[1]:

```
"import os, shutil\n\nval_dir_pos = 'aclImdb/val/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file in os.listdir(val_dir_pos):\n    val =
val_dir_pos +'/'+ file\n    train = train_dir_pos + '/' + file\n
shutil.move(val, train)\n    \nval_dir_neg = 'aclImdb/val/neg'\ntrain_dir_neg =
'aclImdb/train/neg'\n\nfor file in os.listdir(val_dir_neg):\n    val =
val_dir_neg + '/' + file\n    train = train_dir_neg + '/' + file\n
shutil.move(val, train)"
```

0.1.2 Preparing the data

[2]:

```
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.
```

0.1.3 Preparing integer sequence datasets

[3]:

```
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)
```

0.1.4 Embedding layer model building from scratch

[4]:

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

[5]:

```
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"

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

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

_________________________________________________________________
bidirectional (Bidirectional (None, 64)               73984

_________________________________________________________________
dropout (Dropout)           (None, 64)                0

_________________________________________________________________
dense (Dense)               (None, 1)                 65
=================================================================
Total params: 5,194,049
Trainable params: 5,194,049
Non-trainable params: 0

_________________________________________________________________
Epoch 1/10
625/625 [==============================] - 30s 40ms/step - loss: 0.4852 -
accuracy: 0.7793 - val_loss: 0.3640 - val_accuracy: 0.8578
Epoch 2/10
625/625 [==============================] - 25s 39ms/step - loss: 0.3080 -
accuracy: 0.8861 - val_loss: 0.4552 - val_accuracy: 0.8586
```

```
Epoch 3/10
625/625 [==============================] - 25s 40ms/step - loss: 0.2486 -
accuracy: 0.9106 - val_loss: 0.3426 - val_accuracy: 0.8592
Epoch 4/10
625/625 [==============================] - 25s 39ms/step - loss: 0.2096 -
accuracy: 0.9293 - val_loss: 0.3165 - val_accuracy: 0.8730
Epoch 5/10
625/625 [==============================] - 25s 39ms/step - loss: 0.1789 -
accuracy: 0.9385 - val_loss: 0.3498 - val_accuracy: 0.8742
Epoch 6/10
625/625 [==============================] - 25s 39ms/step - loss: 0.1546 -
accuracy: 0.9507 - val_loss: 0.3675 - val_accuracy: 0.8710
Epoch 7/10
625/625 [==============================] - 25s 39ms/step - loss: 0.1232 -
accuracy: 0.9599 - val_loss: 0.4367 - val_accuracy: 0.8826
Epoch 8/10
625/625 [==============================] - 25s 40ms/step - loss: 0.1024 -
accuracy: 0.9691 - val_loss: 0.3835 - val_accuracy: 0.8800
Epoch 9/10
625/625 [==============================] - 25s 40ms/step - loss: 0.0871 -
accuracy: 0.9732 - val_loss: 0.4360 - val_accuracy: 0.8756
Epoch 10/10
625/625 [==============================] - 25s 40ms/step - loss: 0.0741 -
accuracy: 0.9777 - val_loss: 0.4856 - val_accuracy: 0.8706
782/782 [==============================] - 14s 17ms/step - loss: 0.3574 -
accuracy: 0.8570
Test acc: 0.857
```

0.1.5 Embedding layer model building with masking enabled

[6]:

```
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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, None)]            0
_________________________________________________________________
embedding_2 (Embedding)      (None, None, 256)         5120000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                73984
_________________________________________________________________
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 [==============================] - 30s 40ms/step - loss: 0.3992 -
accuracy: 0.8223 - val_loss: 0.2784 - val_accuracy: 0.8890
Epoch 2/10
625/625 [==============================] - 23s 37ms/step - loss: 0.2313 -
accuracy: 0.9112 - val_loss: 0.2610 - val_accuracy: 0.8904
Epoch 3/10
625/625 [==============================] - 23s 37ms/step - loss: 0.1657 -
accuracy: 0.9386 - val_loss: 0.2912 - val_accuracy: 0.8822
Epoch 4/10
625/625 [==============================] - 23s 37ms/step - loss: 0.1220 -
accuracy: 0.9561 - val_loss: 0.3408 - val_accuracy: 0.8594
Epoch 5/10
625/625 [==============================] - 23s 37ms/step - loss: 0.0937 -
accuracy: 0.9677 - val_loss: 0.3690 - val_accuracy: 0.8858
Epoch 6/10
625/625 [==============================] - 23s 37ms/step - loss: 0.0704 -
accuracy: 0.9760 - val_loss: 0.3971 - val_accuracy: 0.8808
Epoch 7/10
625/625 [==============================] - 23s 37ms/step - loss: 0.0488 -
accuracy: 0.9843 - val_loss: 0.4493 - val_accuracy: 0.8832
Epoch 8/10
625/625 [==============================] - 23s 37ms/step - loss: 0.0332 -
accuracy: 0.9887 - val_loss: 0.4954 - val_accuracy: 0.8806
```

```
Epoch 9/10
625/625 [==============================] - 23s 37ms/step - loss: 0.0260 -
accuracy: 0.9919 - val_loss: 0.5148 - val_accuracy: 0.8812
Epoch 10/10
625/625 [==============================] - 23s 37ms/step - loss: 0.0182 -
accuracy: 0.9941 - val_loss: 0.5982 - val_accuracy: 0.8746
782/782 [==============================] - 13s 15ms/step - loss: 0.2978 -
accuracy: 0.8738
Test acc: 0.874
```

Question_5_2

November 20, 2023

0.1 Question_5_2 20,000 training with pretrained word embedding

0.1.1 Dataset preparation

[1]:
```
import os, pathlib, shutil, random
from tensorflow import keras
import tensorflow as tf

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
)   # change the train data directory
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.
```

[2]:

```
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)
```

0.1.2 Download the glove word embeddings

[3]:

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

```
--2023-11-19 18:25:38--  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]
--2023-11-19 18:25:39--  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]
--2023-11-19 18:25:39--  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’

100%[===================================>] 862,182,613 4.43MB/s   in 2m 40s
```

```
2023-11-19 18:28:20 (5.13 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

0.1.3 Parsing the GloVe word-embeddings file

[4]:

```
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.
```

0.1.4 Preparing the GloVe word-embeddings matrix

[5]:

```
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,
)
```

0.1.5 Model that uses a pretrained Embedding layer

[6]:

```
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"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None)]            0
_________________________________________________________________
embedding (Embedding)        (None, None, 100)         2000000
_________________________________________________________________
bidirectional (Bidirectional (None, 64)                34048
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 2,034,113
Trainable params: 34,113
Non-trainable params: 2,000,000
_________________________________________________________________
Epoch 1/10
625/625 [==============================] - 32s 41ms/step - loss: 0.5734 -
accuracy: 0.6947 - val_loss: 0.5154 - val_accuracy: 0.7422
Epoch 2/10
625/625 [==============================] - 24s 38ms/step - loss: 0.4541 -
accuracy: 0.7933 - val_loss: 0.4090 - val_accuracy: 0.8160
Epoch 3/10
625/625 [==============================] - 24s 38ms/step - loss: 0.4059 -
accuracy: 0.8204 - val_loss: 0.3660 - val_accuracy: 0.8388
Epoch 4/10
625/625 [==============================] - 24s 38ms/step - loss: 0.3730 -
accuracy: 0.8413 - val_loss: 0.3598 - val_accuracy: 0.8426
Epoch 5/10
```

```
625/625 [==============================] - 24s 38ms/step - loss: 0.3484 -
accuracy: 0.8530 - val_loss: 0.3452 - val_accuracy: 0.8540
Epoch 6/10
625/625 [==============================] - 24s 38ms/step - loss: 0.3237 -
accuracy: 0.8656 - val_loss: 0.3262 - val_accuracy: 0.8630
Epoch 7/10
625/625 [==============================] - 24s 38ms/step - loss: 0.3092 -
accuracy: 0.8730 - val_loss: 0.3133 - val_accuracy: 0.8682
Epoch 8/10
625/625 [==============================] - 24s 38ms/step - loss: 0.2939 -
accuracy: 0.8796 - val_loss: 0.3023 - val_accuracy: 0.8782
Epoch 9/10
625/625 [==============================] - 24s 38ms/step - loss: 0.2752 -
accuracy: 0.8877 - val_loss: 0.3071 - val_accuracy: 0.8748
Epoch 10/10
625/625 [==============================] - 24s 38ms/step - loss: 0.2628 -
accuracy: 0.8931 - val_loss: 0.2936 - val_accuracy: 0.8820
782/782 [==============================] - 15s 16ms/step - loss: 0.2905 -
accuracy: 0.8761
Test acc: 0.876
```

Question_5_3

November 20, 2023

0.1 Question_5_3 15,000 training with embedding layer

0.1.1 Reorganize the dataset

[1]:
```
'''import os, shutil

val_dir_pos = 'aclImdb/val/pos'
train_dir_pos = 'aclImdb/train/pos'

for file in os.listdir(val_dir_pos):
    val = val_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(val, train)

val_dir_neg = 'aclImdb/val/neg'
train_dir_neg = 'aclImdb/train/neg'

for file in os.listdir(val_dir_neg):
    val = val_dir_neg + '/' + file
    train = train_dir_neg + '/' + file
    shutil.move(val, train)'''
```

[1]:
```
"import os, shutil\n\nval_dir_pos = 'aclImdb/val/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file in os.listdir(val_dir_pos):\n    val =
val_dir_pos +'/'+ file\n    train = train_dir_pos + '/' + file\n
shutil.move(val, train)\n    \nval_dir_neg = 'aclImdb/val/neg'\ntrain_dir_neg =
'aclImdb/train/neg'\n\nfor file in os.listdir(val_dir_neg):\n    val =
val_dir_neg + '/' + file\n    train = train_dir_neg + '/' + file\n
shutil.move(val, train)"
```

0.1.2 Preparing the data

[2]:
```
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(0.4 * 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 15000 files belonging to 2 classes.
Found 10000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

0.1.3 Preparing integer sequence datasets

[3]:

```
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)
```

0.1.4 Embedding layer model building from scratch

[4]:

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

[5]:

```
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_detrain.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_detrain.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model"

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

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

_________________________________________________________________
bidirectional (Bidirectional (None, 64)                73984

_________________________________________________________________
dropout (Dropout)            (None, 64)                0

_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 5,194,049
Trainable params: 5,194,049
Non-trainable params: 0

_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 26s 48ms/step - loss: 0.5250 -
accuracy: 0.7445 - val_loss: 0.4734 - val_accuracy: 0.7947
Epoch 2/10
469/469 [==============================] - 21s 45ms/step - loss: 0.3327 -
accuracy: 0.8736 - val_loss: 0.3403 - val_accuracy: 0.8611
```

```
Epoch 3/10
469/469 [==============================] - 22s 46ms/step - loss: 0.2515 -
accuracy: 0.9119 - val_loss: 0.3305 - val_accuracy: 0.8700
Epoch 4/10
469/469 [==============================] - 22s 46ms/step - loss: 0.1943 -
accuracy: 0.9303 - val_loss: 0.3693 - val_accuracy: 0.8629
Epoch 5/10
469/469 [==============================] - 22s 47ms/step - loss: 0.1620 -
accuracy: 0.9430 - val_loss: 0.4676 - val_accuracy: 0.8510
Epoch 6/10
469/469 [==============================] - 22s 46ms/step - loss: 0.1313 -
accuracy: 0.9581 - val_loss: 0.3755 - val_accuracy: 0.8657
Epoch 7/10
469/469 [==============================] - 22s 46ms/step - loss: 0.1094 -
accuracy: 0.9653 - val_loss: 0.4294 - val_accuracy: 0.8605
Epoch 8/10
469/469 [==============================] - 22s 46ms/step - loss: 0.0943 -
accuracy: 0.9723 - val_loss: 0.4606 - val_accuracy: 0.8581
Epoch 9/10
469/469 [==============================] - 22s 46ms/step - loss: 0.0783 -
accuracy: 0.9771 - val_loss: 0.5311 - val_accuracy: 0.8526
Epoch 10/10
469/469 [==============================] - 22s 47ms/step - loss: 0.0627 -
accuracy: 0.9809 - val_loss: 0.5050 - val_accuracy: 0.8699
782/782 [==============================] - 13s 16ms/step - loss: 0.3835 -
accuracy: 0.8406
Test acc: 0.841
```

0.1.5 Embedding layer model building with masking enabled

[6]:

```
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_detrain.
 ↪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_detrain.
 ↪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_2 (Embedding)      (None, None, 256)         5120000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                73984
_________________________________________________________________
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
469/469 [==============================] - 27s 47ms/step - loss: 0.4390 -
accuracy: 0.7914 - val_loss: 0.2945 - val_accuracy: 0.8777
Epoch 2/10
469/469 [==============================] - 20s 43ms/step - loss: 0.2520 -
accuracy: 0.9019 - val_loss: 0.2947 - val_accuracy: 0.8783
Epoch 3/10
469/469 [==============================] - 20s 44ms/step - loss: 0.1741 -
accuracy: 0.9375 - val_loss: 0.4056 - val_accuracy: 0.8420
Epoch 4/10
469/469 [==============================] - 20s 43ms/step - loss: 0.1231 -
accuracy: 0.9568 - val_loss: 0.3500 - val_accuracy: 0.8822
Epoch 5/10
469/469 [==============================] - 20s 43ms/step - loss: 0.0906 -
accuracy: 0.9687 - val_loss: 0.3852 - val_accuracy: 0.8785
Epoch 6/10
469/469 [==============================] - 20s 43ms/step - loss: 0.0639 -
accuracy: 0.9789 - val_loss: 0.4697 - val_accuracy: 0.8604
Epoch 7/10
469/469 [==============================] - 20s 43ms/step - loss: 0.0470 -
accuracy: 0.9856 - val_loss: 0.4737 - val_accuracy: 0.8624
Epoch 8/10
469/469 [==============================] - 21s 44ms/step - loss: 0.0338 -
```

```
accuracy: 0.9901 - val_loss: 0.6086 - val_accuracy: 0.8664
Epoch 9/10
469/469 [==============================] - 20s 43ms/step - loss: 0.0258 -
accuracy: 0.9912 - val_loss: 0.5645 - val_accuracy: 0.8644
Epoch 10/10
469/469 [==============================] - 20s 43ms/step - loss: 0.0209 -
accuracy: 0.9938 - val_loss: 0.6176 - val_accuracy: 0.8694
782/782 [==============================] - 14s 15ms/step - loss: 0.3133 -
accuracy: 0.8672
Test acc: 0.867
```

Question_5_4

November 20, 2023

0.1 Question_5_4 22,500 training with embedding layer

0.1.1 Reorganize the dataset

[1]:
```
'''import os, shutil

val_dir_pos = 'aclImdb/val/pos'
train_dir_pos = 'aclImdb/train/pos'

for file in os.listdir(val_dir_pos):
    val = val_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(val, train)

val_dir_neg = 'aclImdb/val/neg'
train_dir_neg = 'aclImdb/train/neg'

for file in os.listdir(val_dir_neg):
    val = val_dir_neg + '/' + file
    train = train_dir_neg + '/' + file
    shutil.move(val, train)'''
```

[1]:
```
"import os, shutil\n\nval_dir_pos = 'aclImdb/val/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file in os.listdir(val_dir_pos):\n    val =
val_dir_pos +'/'+ file\n    train = train_dir_pos + '/' + file\n
shutil.move(val, train)\n    \nval_dir_neg = 'aclImdb/val/neg'\ntrain_dir_neg =
'aclImdb/train/neg'\n\nfor file in os.listdir(val_dir_neg):\n    val =
val_dir_neg + '/' + file\n    train = train_dir_neg + '/' + file\n
shutil.move(val, train)"
```

0.1.2 Preparing the data

[2]:
```
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(0.1 * 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 22500 files belonging to 2 classes.
Found 2500 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

0.1.3 Preparing integer sequence datasets

[3]:

```
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)
```

0.1.4 Embedding layer model building from scratch

[4]:

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

[5]:

```
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_intrain.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_intrain.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None)]            0
_________________________________________________________________
embedding_1 (Embedding)      (None, None, 256)         5120000
_________________________________________________________________
bidirectional (Bidirectional (None, 64)                73984
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 5,194,049
Trainable params: 5,194,049
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
704/704 [==============================] - 30s 38ms/step - loss: 0.4725 -
accuracy: 0.7883 - val_loss: 0.3106 - val_accuracy: 0.8832
Epoch 2/10
704/704 [==============================] - 26s 37ms/step - loss: 0.3047 -
accuracy: 0.8890 - val_loss: 0.2784 - val_accuracy: 0.8912
```

```
Epoch 3/10
704/704 [==============================] - 26s 37ms/step - loss: 0.2424 -
accuracy: 0.9142 - val_loss: 0.3334 - val_accuracy: 0.8700
Epoch 4/10
704/704 [==============================] - 26s 37ms/step - loss: 0.2048 -
accuracy: 0.9311 - val_loss: 0.2945 - val_accuracy: 0.8880
Epoch 5/10
704/704 [==============================] - 26s 37ms/step - loss: 0.1707 -
accuracy: 0.9428 - val_loss: 0.4006 - val_accuracy: 0.8540
Epoch 6/10
704/704 [==============================] - 26s 37ms/step - loss: 0.1503 -
accuracy: 0.9513 - val_loss: 0.4193 - val_accuracy: 0.8752
Epoch 7/10
704/704 [==============================] - 26s 37ms/step - loss: 0.1210 -
accuracy: 0.9611 - val_loss: 0.3764 - val_accuracy: 0.8832
Epoch 8/10
704/704 [==============================] - 27s 38ms/step - loss: 0.1028 -
accuracy: 0.9682 - val_loss: 0.3733 - val_accuracy: 0.8884
Epoch 9/10
704/704 [==============================] - 27s 38ms/step - loss: 0.0818 -
accuracy: 0.9741 - val_loss: 0.4033 - val_accuracy: 0.8740
Epoch 10/10
704/704 [==============================] - 27s 38ms/step - loss: 0.0669 -
accuracy: 0.9798 - val_loss: 0.4644 - val_accuracy: 0.8844
782/782 [==============================] - 14s 17ms/step - loss: 0.3239 -
accuracy: 0.8722
Test acc: 0.872
```

0.1.5 Embedding layer model building with masking enabled

[6]:

```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="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_intrain.
 ↪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_intrain.
 ↪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_2 (Embedding)      (None, None, 256)         5120000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                73984
_________________________________________________________________
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
704/704 [==============================] - 32s 37ms/step - loss: 0.3799 -
accuracy: 0.8328 - val_loss: 0.2587 - val_accuracy: 0.8916
Epoch 2/10
704/704 [==============================] - 25s 35ms/step - loss: 0.2296 -
accuracy: 0.9112 - val_loss: 0.3617 - val_accuracy: 0.8800
Epoch 3/10
704/704 [==============================] - 25s 35ms/step - loss: 0.1714 -
accuracy: 0.9366 - val_loss: 0.2803 - val_accuracy: 0.8980
Epoch 4/10
704/704 [==============================] - 25s 35ms/step - loss: 0.1266 -
accuracy: 0.9550 - val_loss: 0.3518 - val_accuracy: 0.8824
Epoch 5/10
704/704 [==============================] - 25s 35ms/step - loss: 0.1015 -
accuracy: 0.9644 - val_loss: 0.3203 - val_accuracy: 0.8828
Epoch 6/10
704/704 [==============================] - 25s 35ms/step - loss: 0.0721 -
accuracy: 0.9759 - val_loss: 0.3958 - val_accuracy: 0.8804
Epoch 7/10
704/704 [==============================] - 25s 35ms/step - loss: 0.0536 -
accuracy: 0.9821 - val_loss: 0.4182 - val_accuracy: 0.8800
Epoch 8/10
704/704 [==============================] - 25s 35ms/step - loss: 0.0374 -
```

```
accuracy: 0.9872 - val_loss: 0.5612 - val_accuracy: 0.8568
Epoch 9/10
704/704 [==============================] - 25s 35ms/step - loss: 0.0267 -
accuracy: 0.9911 - val_loss: 0.5322 - val_accuracy: 0.8664
Epoch 10/10
704/704 [==============================] - 25s 35ms/step - loss: 0.0188 -
accuracy: 0.9939 - val_loss: 0.5649 - val_accuracy: 0.8808
782/782 [==============================] - 13s 15ms/step - loss: 0.2893 -
accuracy: 0.8802
Test acc: 0.880
```

Question_5_5

November 20, 2023

0.1 Question_5_5 160 training with pretrained word embedding

0.1.1 Dataset preparation

[1]:
```
import os, pathlib, shutil, random
from tensorflow import keras
import tensorflow as tf

batch_size = 32
'''base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "train_q2"
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(0.008 * 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
)   # change the train data directory
val_ds = keras.utils.text_dataset_from_directory(
    "aclImdb/train_q2", 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)'''
```

[1]:
```
'train_ds = keras.utils.text_dataset_from_directory(\n    "aclImdb/train",
batch_size=batch_size\n)   # change the train data directory\nval_ds =
keras.utils.text_dataset_from_directory(\n    "aclImdb/train_q2",
batch_size=batch_size\n)\ntest_ds = keras.utils.text_dataset_from_directory(\n
"aclImdb/test", batch_size=batch_size\n)\n#text_only_train_ds =
train_ds.map(lambda x, y: x)'
```

[2]:

```
'''base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_dir = base_dir / "train_q2"
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_q2", batch_size=batch_size
)   # change the train data directory
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 160 files belonging to 2 classes.
Found 40 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

[3]:

```
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)
```

```python
int_test_ds = test_ds.map(
    lambda x, y: (text_vectorization(x), y),
    num_parallel_calls=4)
```

0.1.2 Download the glove word embeddings

[4]:

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

0.1.3 Parsing the GloVe word-embeddings file

[5]:

```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.
```

0.1.4 Preparing the GloVe word-embeddings matrix

[6]:

```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,
)
```

0.1.5 Model that uses a pretrained Embedding layer

[7]:

```
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_smalltrain.
 ↪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_smalltrain.
 ↪keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None)]            0
_________________________________________________________________
embedding (Embedding)        (None, None, 100)         2000000
_________________________________________________________________
bidirectional (Bidirectional (None, 64)                34048
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 2,034,113
Trainable params: 34,113
Non-trainable params: 2,000,000
_________________________________________________________________
Epoch 1/10
5/5 [==============================] - 8s 521ms/step - loss: 0.7336 - accuracy:
0.4938 - val_loss: 0.6800 - val_accuracy: 0.6500
Epoch 2/10
5/5 [==============================] - 0s 61ms/step - loss: 0.6899 - accuracy:
0.5375 - val_loss: 0.6789 - val_accuracy: 0.6000
```

```
Epoch 3/10
5/5 [==============================] - 0s 65ms/step - loss: 0.6779 - accuracy:
0.5750 - val_loss: 0.6731 - val_accuracy: 0.6000
Epoch 4/10
5/5 [==============================] - 0s 68ms/step - loss: 0.6658 - accuracy:
0.6187 - val_loss: 0.6718 - val_accuracy: 0.6000
Epoch 5/10
5/5 [==============================] - 0s 57ms/step - loss: 0.6674 - accuracy:
0.5813 - val_loss: 0.6619 - val_accuracy: 0.6250
Epoch 6/10
5/5 [==============================] - 0s 64ms/step - loss: 0.6224 - accuracy:
0.6562 - val_loss: 0.6576 - val_accuracy: 0.6250
Epoch 7/10
5/5 [==============================] - 0s 63ms/step - loss: 0.6200 - accuracy:
0.6500 - val_loss: 0.6552 - val_accuracy: 0.6250
Epoch 8/10
5/5 [==============================] - 0s 61ms/step - loss: 0.6167 - accuracy:
0.6313 - val_loss: 0.6450 - val_accuracy: 0.6500
Epoch 9/10
5/5 [==============================] - 0s 62ms/step - loss: 0.5925 - accuracy:
0.6938 - val_loss: 0.6544 - val_accuracy: 0.5500
Epoch 10/10
5/5 [==============================] - 0s 55ms/step - loss: 0.5794 - accuracy:
0.7750 - val_loss: 0.6412 - val_accuracy: 0.5750
782/782 [==============================] - 15s 16ms/step - loss: 0.6831 -
accuracy: 0.5732
Test acc: 0.573
```

Question_5_6

November 20, 2023

0.1 Question_5_6 160 training with embedding layer

0.1.1 Preparing the data

[1]:

```
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(0.1 * 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_q2", 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 160 files belonging to 2 classes.
Found 40 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

0.1.2 Preparing integer sequence datasets

[2]:

```
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)
```

0.1.3 Embedding layer model building from scratch

[3]:

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

[4]:

```
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_smalltrain.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_smalltrain.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None)]            0
_________________________________________________________________
embedding_1 (Embedding)      (None, None, 256)         5120000
_________________________________________________________________
bidirectional (Bidirectional (None, 64)                73984
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 5,194,049
Trainable params: 5,194,049
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
5/5 [==============================] - 4s 227ms/step - loss: 0.6955 - accuracy:
0.4688 - val_loss: 0.6946 - val_accuracy: 0.4500
Epoch 2/10
5/5 [==============================] - 0s 93ms/step - loss: 0.6755 - accuracy:
0.6687 - val_loss: 0.6945 - val_accuracy: 0.5500
Epoch 3/10
5/5 [==============================] - 0s 83ms/step - loss: 0.6422 - accuracy:
0.7812 - val_loss: 0.6894 - val_accuracy: 0.5250
Epoch 4/10
5/5 [==============================] - 0s 84ms/step - loss: 0.5474 - accuracy:
0.8188 - val_loss: 0.6898 - val_accuracy: 0.5250
Epoch 5/10
5/5 [==============================] - 0s 79ms/step - loss: 0.4489 - accuracy:
0.9375 - val_loss: 0.6408 - val_accuracy: 0.5750
Epoch 6/10
5/5 [==============================] - 1s 94ms/step - loss: 0.2812 - accuracy:
0.9812 - val_loss: 0.6212 - val_accuracy: 0.6750
Epoch 7/10
5/5 [==============================] - 0s 68ms/step - loss: 0.2113 - accuracy:
0.9812 - val_loss: 0.6469 - val_accuracy: 0.6500
Epoch 8/10
5/5 [==============================] - 0s 91ms/step - loss: 0.2887 - accuracy:
0.9125 - val_loss: 0.6402 - val_accuracy: 0.6750
Epoch 9/10
5/5 [==============================] - 0s 88ms/step - loss: 0.1542 - accuracy:
1.0000 - val_loss: 0.6197 - val_accuracy: 0.6750
Epoch 10/10
5/5 [==============================] - 0s 62ms/step - loss: 0.1183 - accuracy:
1.0000 - val_loss: 0.7093 - val_accuracy: 0.6500
```

```
782/782 [==============================] - 14s 17ms/step - loss: 0.6633 -
accuracy: 0.6406
Test acc: 0.641
```

0.1.4 Embedding layer model building with masking enabled

[5]:

```
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_smalltrain.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_smalltrain.
 ↪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_2 (Embedding)     (None, None, 256)         5120000

_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)               73984

_________________________________________________________________
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
5/5 [==============================] - 8s 503ms/step - loss: 0.6921 - accuracy:
0.5500 - val_loss: 0.6932 - val_accuracy: 0.5250
Epoch 2/10
5/5 [==============================] - 0s 66ms/step - loss: 0.6627 - accuracy:
0.8625 - val_loss: 0.6922 - val_accuracy: 0.4750
Epoch 3/10
5/5 [==============================] - 0s 71ms/step - loss: 0.6076 - accuracy:
0.9438 - val_loss: 0.6864 - val_accuracy: 0.6250
Epoch 4/10
5/5 [==============================] - 0s 62ms/step - loss: 0.4524 - accuracy:
0.9750 - val_loss: 0.6270 - val_accuracy: 0.6500
Epoch 5/10
5/5 [==============================] - 0s 62ms/step - loss: 0.3675 - accuracy:
0.8375 - val_loss: 0.5905 - val_accuracy: 0.6750
Epoch 6/10
5/5 [==============================] - 0s 66ms/step - loss: 0.1127 - accuracy:
1.0000 - val_loss: 0.6084 - val_accuracy: 0.7000
Epoch 7/10
5/5 [==============================] - 0s 65ms/step - loss: 0.0685 - accuracy:
1.0000 - val_loss: 0.5916 - val_accuracy: 0.7500
Epoch 8/10
5/5 [==============================] - 0s 72ms/step - loss: 0.0540 - accuracy:
1.0000 - val_loss: 0.7243 - val_accuracy: 0.6750
Epoch 9/10
5/5 [==============================] - 0s 69ms/step - loss: 0.0348 - accuracy:
1.0000 - val_loss: 0.6582 - val_accuracy: 0.7750
Epoch 10/10
5/5 [==============================] - 0s 76ms/step - loss: 0.0317 - accuracy:
1.0000 - val_loss: 0.6930 - val_accuracy: 0.6750
782/782 [==============================] - 14s 15ms/step - loss: 0.6209 -
accuracy: 0.6546
Test acc: 0.655
```

Question_5_7

November 20, 2023

0.1 Question_5_7 2,2500 training with pretrained word embedding

0.1.1 Dataset preparation

[1]:

```
'''import os, shutil

val_dir_pos = 'aclImdb/val/pos'
train_dir_pos = 'aclImdb/train_q2/pos'

for file in os.listdir(val_dir_pos):
    val = val_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(val, train)

val_dir_neg = 'aclImdb/val/neg'
train_dir_neg = 'aclImdb/train_q2/neg'

for file in os.listdir(val_dir_neg):
    val = val_dir_neg + '/' + file
    train = train_dir_neg + '/' + file
    shutil.move(val, train)'''
```

[1]:

```
"import os, shutil\n\nval_dir_pos = 'aclImdb/val/pos'\ntrain_dir_pos =
'aclImdb/train_q2/pos'\n\nfor file in os.listdir(val_dir_pos):\n    val =
val_dir_pos +'/'+ file\n    train = train_dir_pos + '/' + file\n
shutil.move(val, train)\n    \nval_dir_neg = 'aclImdb/val/neg'\ntrain_dir_neg =
'aclImdb/train_q2/neg'\n\nfor file in os.listdir(val_dir_neg):\n    val =
val_dir_neg + '/' + file\n    train = train_dir_neg + '/' + file\n
shutil.move(val, train)"
```

[2]:

```
'''train_q2_dir_pos = 'aclImdb/train_q2/pos'
train_dir_pos = 'aclImdb/train/pos'

for file in os.listdir(train_q2_dir_pos):
    small_train = train_q2_dir_pos +'/'+ file
    train = train_dir_pos + '/' + file
    shutil.move(small_train, train)
```

```
train_q2_dir_neg = 'aclImdb/train_q2/neg'
train_dir_neg = 'aclImdb/train/neg'

for file in os.listdir(train_q2_dir_neg):
    small_train = train_q2_dir_neg +'/'+ file
    train = train_dir_neg + '/' + file
    shutil.move(small_train, train)'''
```

[2]:
```
"train_q2_dir_pos = 'aclImdb/train_q2/pos'\ntrain_dir_pos =
'aclImdb/train/pos'\n\nfor file in os.listdir(train_q2_dir_pos):\n
small_train = train_q2_dir_pos +'/'+ file\n    train = train_dir_pos + '/' +
file\n    shutil.move(small_train, train)\n    \ntrain_q2_dir_neg =
'aclImdb/train_q2/neg'\ntrain_dir_neg = 'aclImdb/train/neg'\n\nfor file in
os.listdir(train_q2_dir_neg):\n    small_train = train_q2_dir_neg +'/'+ file\n
train = train_dir_neg + '/' + file\n    shutil.move(small_train, train)"
```

[3]:
```
import os, pathlib, shutil, random
from tensorflow import keras
import tensorflow as tf

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(0.1 * 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
)   # change the train data directory
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 22500 files belonging to 2 classes.
Found 2500 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

[4]:

```python
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)
```

0.1.2 Download the glove word embeddings

[5]:

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

0.1.3 Parsing the GloVe word-embeddings file

[6]:

```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.
```

0.1.4 Preparing the GloVe word-embeddings matrix

[7]:
```
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,
)
```

0.1.5 Model that uses a pretrained Embedding layer

[8]:
```
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_intrain.
 ↪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_intrain.keras")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
```

```
Model: "model"
_________________________________________________________________
```

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

-----------------------------------------------------------------
embedding (Embedding)        (None, None, 100)         2000000

-----------------------------------------------------------------
bidirectional (Bidirectional (None, 64)                34048

-----------------------------------------------------------------
dropout (Dropout)            (None, 64)                0

-----------------------------------------------------------------
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 2,034,113
Trainable params: 34,113
Non-trainable params: 2,000,000

-----------------------------------------------------------------
Epoch 1/10
704/704 [==============================] - 33s 38ms/step - loss: 0.5665 -
accuracy: 0.7031 - val_loss: 0.4758 - val_accuracy: 0.7672
Epoch 2/10
704/704 [==============================] - 25s 35ms/step - loss: 0.4425 -
accuracy: 0.7979 - val_loss: 0.4016 - val_accuracy: 0.8208
Epoch 3/10
704/704 [==============================] - 25s 35ms/step - loss: 0.3901 -
accuracy: 0.8305 - val_loss: 0.5412 - val_accuracy: 0.7540
Epoch 4/10
704/704 [==============================] - 25s 35ms/step - loss: 0.3558 -
accuracy: 0.8476 - val_loss: 0.5107 - val_accuracy: 0.7748
Epoch 5/10
704/704 [==============================] - 25s 35ms/step - loss: 0.3316 -
accuracy: 0.8612 - val_loss: 0.3157 - val_accuracy: 0.8676
Epoch 6/10
704/704 [==============================] - 25s 35ms/step - loss: 0.3080 -
accuracy: 0.8721 - val_loss: 0.3686 - val_accuracy: 0.8476
Epoch 7/10
704/704 [==============================] - 25s 35ms/step - loss: 0.2915 -
accuracy: 0.8782 - val_loss: 0.3551 - val_accuracy: 0.8576
Epoch 8/10
704/704 [==============================] - 25s 35ms/step - loss: 0.2740 -
accuracy: 0.8877 - val_loss: 0.3211 - val_accuracy: 0.8688
Epoch 9/10
704/704 [==============================] - 25s 35ms/step - loss: 0.2616 -
accuracy: 0.8952 - val_loss: 0.3237 - val_accuracy: 0.8624
Epoch 10/10
704/704 [==============================] - 25s 35ms/step - loss: 0.2454 -
accuracy: 0.9008 - val_loss: 0.3294 - val_accuracy: 0.8700
782/782 [==============================] - 14s 16ms/step - loss: 0.3124 -
accuracy: 0.8655
```

```
Test acc: 0.865
```