Text and Sequence

May 2, 2024

1 Text and Sequence Assignment

1.1 Author : Dev, Sakshi

We will be using IMDB data for this text and sequence problem. Firstly, we need to create a validation set with 80% of training dataset and setting apart 20% for training.

1.1.1 Reading Data

[1]:
```
import os, shutil , pathlib , random
base_dir = pathlib.Path("/Users/devmarwah/Documents/MSBA assignments/Advanced␣
 ↪Machine Learning/Text and Sequencing/IMDB data")
val_dir= base_dir/"validation"
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 = 5000
    val_file = files[:num_val_samples]
    for fname in val_file:
        shutil.move(train_dir/category/fname,
                    val_dir/category,fname)
```

Making a small training sample as well :

[2]:
```
train_dir_1 =base_dir/"train1"
for category in ("neg","pos"):
    os.makedirs(train_dir_1/category)
    files= os.listdir(train_dir/category)
    random.Random(1337).shuffle(files)
    num_train_samples = 50
    train_file = files[:num_train_samples]
    for fname in train_file:
        shutil.move(train_dir/category/fname,
                    train_dir_1/category,fname)
```

Reading our datasets :

[3]:
```
from tensorflow import keras
batch_size = 32
train = keras.utils.
 ↪text_dataset_from_directory(train_dir_1,batch_size=batch_size)
validation=keras.utils.
 ↪text_dataset_from_directory(val_dir,batch_size=batch_size)
test=keras.utils.text_dataset_from_directory(base_dir/
 ↪"test",batch_size=batch_size)
```

```
Found 100 files belonging to 2 classes.
Found 10000 files belonging to 2 classes.
Found 25000 files belonging to 2 classes.
```

1.1.2 Trying sequencing model

Preparing dataset for this model :

[4]:
```
from tensorflow.keras import layers
max_length = 150 # Cutting off values after 150 words
max_tokens = 10000 # Considering only top 10,000 words
text_vectorization = layers.TextVectorization(
    max_tokens=max_tokens,
    output_mode="int",
    output_sequence_length=max_length,
)
text_only_train_ds = train.map(lambda x, y: x)
# Turning text to vectors
text_vectorization.adapt(text_only_train_ds)
int_train_ds = train.map(
lambda x, y: (text_vectorization(x), y), num_parallel_calls=4)
int_val_ds = validation.map(
lambda x, y: (text_vectorization(x), y), num_parallel_calls=4)
int_test_ds = test.map(
lambda x, y: (text_vectorization(x), y), num_parallel_calls=4)
```

```
2024-04-27 13:19:52.616018: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

Model Construction - Embedding Layer

[5]:
```
 import tensorflow as tf
inputs=keras.Input(shape=(None,), dtype="int64")
embedded= layers.Embedding(input_dim=max_tokens, output_dim=256,␣
 ↪mask_zero=True)(inputs)
# We have turned mask on because training bi-directional LSTM can take longer␣
 ↪time
x= layers.Bidirectional(layers.LSTM(32))(embedded)
x=layers.Dropout(0.5)(x)
outputs= layers.Dense(1,activation="sigmoid")(x)
```

[6]:
```
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
callbacks = [
    keras.callbacks.ModelCheckpoint("/Users/devmarwah/Documents/MSBA␣
 ↪assignments/Advanced Machine Learning/Text and Sequencing/one_hot_bidir_lstm.
 ↪keras",
                                    save_best_only=True)
]
model.summary()
```

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, None)	0	-
embedding (Embedding)	(None, None, 256)	2,560,000	input_layer[0][0]
not_equal (NotEqual)	(None, None)	0	input_layer[0][0]
bidirectional (Bidirectional)	(None, 64)	73,984	embedding[0][0], not_equal[0][0]
dropout (Dropout)	(None, 64)	0	bidirectional[0]…
dense (Dense)	(None, 1)	65	dropout[0][0]

Total params: 2,634,049 (10.05 MB)

Trainable params: 2,634,049 (10.05 MB)

Non-trainable params: 0 (0.00 B)

Fitting the model on our testing dataset

[7]:
```
model.fit(int_train_ds,
          validation_data=int_val_ds,
          epochs=10,
```

```
        callbacks=callbacks)
```

```
Epoch 1/10
4/4             6s 2s/step -
accuracy: 0.4818 - loss: 0.6944 - val_accuracy: 0.5032 - val_loss: 0.6934
Epoch 2/10
4/4             5s 2s/step -
accuracy: 0.7362 - loss: 0.6836 - val_accuracy: 0.5099 - val_loss: 0.6930
Epoch 3/10
4/4             5s 2s/step -
accuracy: 0.8605 - loss: 0.6745 - val_accuracy: 0.5021 - val_loss: 0.6929
Epoch 4/10
4/4             5s 2s/step -
accuracy: 0.8587 - loss: 0.6659 - val_accuracy: 0.5110 - val_loss: 0.6927
Epoch 5/10
4/4             8s 3s/step -
accuracy: 0.7935 - loss: 0.6541 - val_accuracy: 0.5189 - val_loss: 0.6922
Epoch 6/10
4/4             8s 3s/step -
accuracy: 0.9400 - loss: 0.6336 - val_accuracy: 0.5178 - val_loss: 0.6921
Epoch 7/10
4/4             8s 3s/step -
accuracy: 0.9265 - loss: 0.6022 - val_accuracy: 0.5318 - val_loss: 0.6906
Epoch 8/10
4/4             8s 3s/step -
accuracy: 0.9448 - loss: 0.5635 - val_accuracy: 0.5279 - val_loss: 0.6902
Epoch 9/10
4/4             8s 3s/step -
accuracy: 0.9569 - loss: 0.5023 - val_accuracy: 0.5012 - val_loss: 0.7443
Epoch 10/10
4/4             8s 3s/step -
accuracy: 0.8550 - loss: 0.4527 - val_accuracy: 0.5146 - val_loss: 0.7370
```

[7]:
```
<keras.src.callbacks.history.History at 0x1766aec10>
```

Testing this model

[8]:
```
print("\n Model's accuracy:",round(model.evaluate(int_test_ds)[1]*100,2),"%")
```

```
782/782             18s 23ms/step -
accuracy: 0.5164 - loss: 0.7372

 Model's accuracy: 51.42 %
```

Hence, our first model's accuracy with LSTM and embedding is just **51.42%** which is quite low. We will now try a pre-trained word embedding.

Model Construction - Pretrained word embedded

Parsing after downloading the glove pretrained word-embedding.

[9]:
```
import numpy as np
path_to_glove_file = "/Users/devmarwah/Documents/MSBA assignments/Advanced␣
 ↪Machine Learning/Text and Sequencing/glove.6B/glove.6B.100d.txt"

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

Preparing a matrix of GloVe :

[10]:
```
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
```

Making an embedding layer with this embedded matrix :

[11]:
```
embedding_layer= layers.Embedding(max_tokens,
                                  embedding_dim,
                                  embeddings_initializer=keras.initializers.
 ↪Constant(embedding_matrix),
                                  trainable=False,
                                  mask_zero=True)
```

Making a final model with pretrained work-embedding :

[12]:
```
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("/Users/devmarwah/Documents/MSBA␣
 ↪assignments/Advanced Machine Learning/Text and Sequencing/
 ↪glove_embeddings_sequence_model.keras",
                                    save_best_only=True)
```

```
]
```

```
Model: "functional_3"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, None)	0	-
embedding_1 (Embedding)	(None, None, 100)	1,000,000	input_layer_1[0]…
not_equal_1 (NotEqual)	(None, None)	0	input_layer_1[0]…
bidirectional_1 (Bidirectional)	(None, 64)	34,048	embedding_1[0][0… not_equal_1[0][0]
dropout_1 (Dropout)	(None, 64)	0	bidirectional_1[…
dense_1 (Dense)	(None, 1)	65	dropout_1[0][0]

```
Total params: 1,034,113 (3.94 MB)

Trainable params: 1,034,113 (3.94 MB)

Non-trainable params: 0 (0.00 B)
```

Training this model on our training dataset :

[13]:
```
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10,
callbacks=callbacks)
```

```
Epoch 1/10
4/4             6s 2s/step -
accuracy: 0.4599 - loss: 0.7884 - val_accuracy: 0.5023 - val_loss: 0.7113
Epoch 2/10
4/4             5s 2s/step -
accuracy: 0.5549 - loss: 0.6722 - val_accuracy: 0.5100 - val_loss: 0.7009
Epoch 3/10
4/4             5s 2s/step -
accuracy: 0.5099 - loss: 0.6993 - val_accuracy: 0.5084 - val_loss: 0.6960
Epoch 4/10
4/4             5s 2s/step -
```

```
accuracy: 0.5520 - loss: 0.6879 - val_accuracy: 0.5189 - val_loss: 0.6929
Epoch 5/10
4/4             5s 2s/step -
accuracy: 0.6635 - loss: 0.6248 - val_accuracy: 0.5227 - val_loss: 0.6922
Epoch 6/10
4/4             5s 2s/step -
accuracy: 0.5994 - loss: 0.6553 - val_accuracy: 0.5264 - val_loss: 0.6912
Epoch 7/10
4/4             5s 2s/step -
accuracy: 0.6260 - loss: 0.6428 - val_accuracy: 0.5284 - val_loss: 0.6954
Epoch 8/10
4/4             5s 2s/step -
accuracy: 0.6689 - loss: 0.6680 - val_accuracy: 0.5079 - val_loss: 0.7074
Epoch 9/10
4/4             5s 2s/step -
accuracy: 0.6194 - loss: 0.6576 - val_accuracy: 0.5259 - val_loss: 0.6979
Epoch 10/10
4/4             5s 2s/step -
accuracy: 0.7012 - loss: 0.5911 - val_accuracy: 0.5364 - val_loss: 0.6909
```

[13]:
```
<keras.src.callbacks.history.History at 0x284cd6fd0>
```

Testing this model on our testing data :

[14]:
```
print("\n Model's Accuracy:",round(model.evaluate(int_test_ds)[1]*100,2))
```

```
782/782             12s 15ms/step -
accuracy: 0.5329 - loss: 0.6908

 Model's Accuracy: 53.69
```

Pre-trained embedding is not really helpful in this case. Hence, training from scratch worked better for this dataset. Now, we will try to increase training sample size and then train our model again.

Increasing training size by 7000 samples

[15]:
```
for category in ("neg","pos"):
    files= os.listdir(train_dir/category)
    random.Random(1337).shuffle(files)
    num_train_samples = 3500
    train_file = files[:num_train_samples]
    for fname in train_file:
        shutil.move(train_dir/category/fname,
                    train_dir_1/category,fname)
```

Making a training dataset again :

[16]:
```
train = keras.utils.
 ↪text_dataset_from_directory(train_dir_1,batch_size=batch_size)
int_train_ds = train.map(
```

```
lambda x, y: (text_vectorization(x), y), num_parallel_calls=4)
```

```
Found 7100 files belonging to 2 classes.
```

Training the last pretrained embedding model with new training dataset :

[17]:
```
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10,
callbacks=callbacks)
```

```
Epoch 1/10
222/222             19s 84ms/step -
accuracy: 0.5677 - loss: 0.6838 - val_accuracy: 0.6883 - val_loss: 0.5900
Epoch 2/10
222/222             20s 91ms/step -
accuracy: 0.6891 - loss: 0.5843 - val_accuracy: 0.7678 - val_loss: 0.4902
Epoch 3/10
222/222             20s 91ms/step -
accuracy: 0.7569 - loss: 0.5130 - val_accuracy: 0.7687 - val_loss: 0.4837
Epoch 4/10
222/222             20s 90ms/step -
accuracy: 0.7942 - loss: 0.4484 - val_accuracy: 0.8093 - val_loss: 0.4346
Epoch 5/10
222/222             20s 88ms/step -
accuracy: 0.8222 - loss: 0.4099 - val_accuracy: 0.8072 - val_loss: 0.4569
Epoch 6/10
222/222             20s 90ms/step -
accuracy: 0.8476 - loss: 0.3613 - val_accuracy: 0.8168 - val_loss: 0.4379
Epoch 7/10
222/222             20s 92ms/step -
accuracy: 0.8602 - loss: 0.3317 - val_accuracy: 0.8075 - val_loss: 0.4466
Epoch 8/10
222/222             20s 90ms/step -
accuracy: 0.8772 - loss: 0.3005 - val_accuracy: 0.7936 - val_loss: 0.5010
Epoch 9/10
222/222             20s 88ms/step -
accuracy: 0.8870 - loss: 0.2743 - val_accuracy: 0.8126 - val_loss: 0.5556
Epoch 10/10
222/222             20s 89ms/step -
accuracy: 0.9015 - loss: 0.2440 - val_accuracy: 0.8118 - val_loss: 0.5086
```

[17]:
```
<keras.src.callbacks.history.History at 0x2d68b8410>
```

Testing the model now :

[18]:
```
print("\n Model's Accuracy:",round(model.evaluate(int_test_ds)[1]*100,2))
```

```
782/782             12s 15ms/step -
accuracy: 0.8226 - loss: 0.4714

 Model's Accuracy: 82.05
```

Increasing samples did not really increase any acuracy.

Increasing training sample again by 7000

[19]:
```
for category in ("neg","pos"):
    files= os.listdir(train_dir/category)
    random.Random(1337).shuffle(files)
    num_train_samples = 3500
    train_file = files[:num_train_samples]
    for fname in train_file:
        shutil.move(train_dir/category/fname,
                    train_dir_1/category,fname)
```

Reading new training set:

[20]:
```
train = keras.utils.
  ↪text_dataset_from_directory(train_dir_1,batch_size=batch_size)
int_train_ds = train.map(
lambda x, y: (text_vectorization(x), y), num_parallel_calls=4)
```

```
Found 14100 files belonging to 2 classes.
```

Training this model again :

[21]:
```
model.fit(int_train_ds, validation_data=int_val_ds, epochs=10)
```

```
Epoch 1/10
441/441             35s 79ms/step -
accuracy: 0.8645 - loss: 0.3277 - val_accuracy: 0.8279 - val_loss: 0.3957
Epoch 2/10
441/441             34s 77ms/step -
accuracy: 0.8832 - loss: 0.2903 - val_accuracy: 0.8309 - val_loss: 0.4049
Epoch 3/10
441/441             34s 77ms/step -
accuracy: 0.8924 - loss: 0.2706 - val_accuracy: 0.8308 - val_loss: 0.4207
Epoch 4/10
441/441             35s 80ms/step -
accuracy: 0.9031 - loss: 0.2490 - val_accuracy: 0.8290 - val_loss: 0.4348
Epoch 5/10
441/441             38s 85ms/step -
accuracy: 0.9088 - loss: 0.2264 - val_accuracy: 0.8125 - val_loss: 0.4916
Epoch 6/10
441/441             36s 82ms/step -
accuracy: 0.9225 - loss: 0.2067 - val_accuracy: 0.8277 - val_loss: 0.4868
Epoch 7/10
441/441             39s 89ms/step -
accuracy: 0.9311 - loss: 0.1781 - val_accuracy: 0.8179 - val_loss: 0.4934
Epoch 8/10
441/441             38s 87ms/step -
accuracy: 0.9386 - loss: 0.1605 - val_accuracy: 0.8258 - val_loss: 0.5785
```

```
Epoch 9/10
441/441             38s 87ms/step -
accuracy: 0.9477 - loss: 0.1361 - val_accuracy: 0.8122 - val_loss: 0.5701
Epoch 10/10
441/441             40s 91ms/step -
accuracy: 0.9593 - loss: 0.1171 - val_accuracy: 0.8180 - val_loss: 0.6679
```

[21]:
```
<keras.src.callbacks.history.History at 0x2d3403910>
```

Testing this model :

[22]:
```
print("\n Model's Accuracy:",round(model.evaluate(int_test_ds)[1]*100,2))
```

```
782/782             13s 17ms/step -
accuracy: 0.8254 - loss: 0.6293

 Model's Accuracy: 82.2
```

There is not any improvement with the second increase of training size. This explains that beyond a certain point, increasing traning set size does not affect accuracy.