Open in Colab

OMAR KILLBI

MPDS2

Sentiment Analysis Using Tenserflow

Deep Learning

```
import tensorflow as tf
print(tf.__version__)

# !pip install -q tensorflow-datasets
```

```
2.7.0
```

```
import tensorflow_datasets as tfds
imdb, info = tfds.load("imdb_reviews", with_info=True, as_supervised=True)
```

```
Downloading and preparing dataset imdb_reviews/plain_text/1.0.0 (download: 80.23 MiB
Dl Completed...: 100%     1/1 [00:02<00:00, 2.72s/ url]

Dl Size...: 100%     80/80 [00:02<00:00, 40.45 MiB/s]


Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/plain_text/
100%                                   24999/25000 [00:00<00:00, 127653.63 examples/s]
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/plain_text/
100%                                   24999/25000 [00:00<00:00, 138056.98 examples/s]
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/plain_text/
100%                                   49999/50000 [00:00<00:00, 160180.64 examples/s]
WARNING:absl:Dataset is using deprecated text encoder API which will be removed soon
Dataset imdb_reviews downloaded and prepared to /root/tensorflow_datasets/imdb_revie
```

```
import numpy as np

train_data, test_data = imdb['train'], imdb['test']

training_sentences = []
training_labels = []

testing_sentences = []
testing_labels = []
```

```
# str(s.tonumpy()) is needed in Python3 instead of just s.numpy()
for s,l in train_data:
  training_sentences.append(s.numpy().decode('utf8'))
  training_labels.append(l.numpy())

for s,l in test_data:
  testing_sentences.append(s.numpy().decode('utf8'))
  testing_labels.append(l.numpy())

training_labels_final = np.array(training_labels)
testing_labels_final = np.array(testing_labels)
```

```
vocab_size = 10000
embedding_dim = 16
max_length = 120
trunc_type='post'
oov_tok = "<OOV>"
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(training_sentences)
padded = pad_sequences(sequences,maxlen=max_length, truncating=trunc_type)

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences,maxlen=max_length)
```

```
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])

def decode_review(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])

print(decode_review(padded[2]))
print(training_sentences[2])
```

```
mann photographs the <OOV> rocky mountains in a superb fashion and jimmy stewart and
Mann photographs the Alberta Rocky Mountains in a superb fashion, and Jimmy Stewart
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 120, 16)           160000

 flatten (Flatten)           (None, 1920)              0

 dense (Dense)               (None, 6)                 11526

 dense_1 (Dense)             (None, 1)                 7

=================================================================
Total params: 171,533
Trainable params: 171,533
Non-trainable params: 0
_________________________________________________________________
```

```
num_epochs = 10
model.fit(padded, training_labels_final, epochs=num_epochs, validation_data=(testing_padde
```

```
Epoch 1/10
782/782 [==============================] - 9s 9ms/step - loss: 0.4905 - accuracy: 0.
Epoch 2/10
782/782 [==============================] - 7s 9ms/step - loss: 0.2360 - accuracy: 0.
Epoch 3/10
782/782 [==============================] - 6s 8ms/step - loss: 0.0863 - accuracy: 0.
Epoch 4/10
782/782 [==============================] - 7s 9ms/step - loss: 0.0214 - accuracy: 0.
Epoch 5/10
782/782 [==============================] - 7s 9ms/step - loss: 0.0057 - accuracy: 0.
Epoch 6/10
782/782 [==============================] - 6s 8ms/step - loss: 0.0018 - accuracy: 1.
Epoch 7/10
782/782 [==============================] - 7s 9ms/step - loss: 8.1942e-04 - accuracy
Epoch 8/10
782/782 [==============================] - 7s 9ms/step - loss: 4.5244e-04 - accuracy
Epoch 9/10
782/782 [==============================] - 6s 8ms/step - loss: 2.6272e-04 - accuracy
Epoch 10/10
782/782 [==============================] - 6s 7ms/step - loss: 1.5730e-04 - accuracy
<keras.callbacks.History at 0x7f34e11a8390>
```

```
e = model.layers[0]
weights = e.get_weights()[0]
print(weights.shape) # shape: (vocab_size, embedding_dim)
```

```
(10000, 16)
```

```
import io

out_v = io.open('vecs.tsv', 'w', encoding='utf-8')
out_m = io.open('meta.tsv', 'w', encoding='utf-8')
```

```
for word_num in range(1, vocab_size):
  word = reverse_word_index[word_num]
  embeddings = weights[word_num]
  out_m.write(word + "\n")
  out_v.write('\t'.join([str(x) for x in embeddings]) + "\n")
out_v.close()
out_m.close()
```

```
try:
  from google.colab import files
except ImportError:
  pass
else:
  files.download('vecs.tsv')
  files.download('meta.tsv')
```

```
sentence = "I really think this is amazing. honest."
sequence = tokenizer.texts_to_sequences([sentence])
print(sequence)
```

```
[[11, 64, 102, 12, 7, 478, 1200]]
```

✓ 0s completed at 9:17 AM