lab-assignment-6-dl-3

October 21, 2024

[2]:
```
from keras.datasets import imdb
```

[5]:
```
vocabulary_size = 5000
# Load the IMDB dataset
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=vocabulary_size)
# Print the number of training and test samples
print('Loaded dataset with {} training samples, {} test samples'.
 ↪format(len(X_train), len(X_test)))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/imdb.npz
17464789/17464789             4s
0us/step
Loaded dataset with 25000 training samples, 25000 test samples
```

[6]:
```
print('---review---')
print(X_train[6])
print('---label---')
print(y_train[6])
```

```
---review---
[1, 2, 365, 1234, 5, 1156, 354, 11, 14, 2, 2, 7, 1016, 2, 2, 356, 44, 4, 1349,
500, 746, 5, 200, 4, 4132, 11, 2, 2, 1117, 1831, 2, 5, 4831, 26, 6, 2, 4183, 17,
369, 37, 215, 1345, 143, 2, 5, 1838, 8, 1974, 15, 36, 119, 257, 85, 52, 486, 9,
6, 2, 2, 63, 271, 6, 196, 96, 949, 4121, 4, 2, 7, 4, 2212, 2436, 819, 63, 47,
77, 2, 180, 6, 227, 11, 94, 2494, 2, 13, 423, 4, 168, 7, 4, 22, 5, 89, 665, 71,
270, 56, 5, 13, 197, 12, 161, 2, 99, 76, 23, 2, 7, 419, 665, 40, 91, 85, 108, 7,
4, 2084, 5, 4773, 81, 55, 52, 1901]
---label---
1
```

[7]:
```
word2id = imdb.get_word_index()
id2word = {i: word for word, i in word2id.items()}
print('---review with words---')
print([id2word.get(i, ' ') for i in X_train[6]])
print('---label---')
print(y_train[6])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/imdb_word_index.json
1641221/1641221             0s
0us/step
---review with words---
['the', 'and', 'full', 'involving', 'to', 'impressive', 'boring', 'this', 'as',
'and', 'and', 'br', 'villain', 'and', 'and', 'need', 'has', 'of', 'costumes',
'b', 'message', 'to', 'may', 'of', 'props', 'this', 'and', 'and', 'concept',
'issue', 'and', 'to', "god's", 'he', 'is', 'and', 'unfolds', 'movie', 'women',
'like', "isn't", 'surely', "i'm", 'and', 'to', 'toward', 'in', "here's", 'for',
'from', 'did', 'having', 'because', 'very', 'quality', 'it', 'is', 'and', 'and',
'really', 'book', 'is', 'both', 'too', 'worked', 'carl', 'of', 'and', 'br',
'of', 'reviewer', 'closer', 'figure', 'really', 'there', 'will', 'and',
'things', 'is', 'far', 'this', 'make', 'mistakes', 'and', 'was', "couldn't",
'of', 'few', 'br', 'of', 'you', 'to', "don't", 'female', 'than', 'place', 'she',
'to', 'was', 'between', 'that', 'nothing', 'and', 'movies', 'get', 'are', 'and',
'br', 'yes', 'female', 'just', 'its', 'because', 'many', 'br', 'of', 'overly',
'to', 'descent', 'people', 'time', 'very', 'bland']
---label---
1
```

[8]:
```
print('Maximum review length: {}'.format(
len(max((X_train + X_test), key=len))))
```

```
Maximum review length: 2697
```

[9]:
```
print('Minimum review length: {}'.format(
len(min((X_test + X_test), key=len))))
```

```
Minimum review length: 14
```

[12]:
```
from keras.preprocessing import sequence  # Correct import
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)
```

[13]:
```
from keras import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout

model = Sequential()
embedding_size = 32
model.add(Embedding(vocabulary_size, embedding_size))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))

print(model.summary())
```

```
Model: "sequential"
```

```
Layer (type)                         Output Shape                  ␣
↪Param #

embedding (Embedding)                ?                             0␣
↪(unbuilt)

lstm (LSTM)                          ?                             0␣
↪(unbuilt)

dense (Dense)                        ?                             0␣
↪(unbuilt)


Total params: 0 (0.00 B)


Trainable params: 0 (0.00 B)


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

None

[14]:
```
model.compile(loss='binary_crossentropy',
optimizer='adam',
metrics=['accuracy'])
```

[16]:
```
batch_size = 64
num_epochs = 3
X_valid, y_valid = X_train[:batch_size], y_train[:batch_size]
X_train2, y_train2 = X_train[batch_size:], y_train[batch_size:]

model.fit(X_train2, y_train2,
          validation_data=(X_valid, y_valid),
          batch_size=batch_size,
          epochs=num_epochs)
```

```
Epoch 1/3
390/390             147s 372ms/step -
accuracy: 0.6638 - loss: 0.5844 - val_accuracy: 0.9062 - val_loss: 0.2588
Epoch 2/3
390/390             170s 436ms/step -
accuracy: 0.8658 - loss: 0.3220 - val_accuracy: 0.9219 - val_loss: 0.1898
Epoch 3/3
390/390             167s 429ms/step -
```

```
accuracy: 0.8923 - loss: 0.2739 - val_accuracy: 0.9375 - val_loss: 0.2227
```

[16]:
```
<keras.src.callbacks.history.History at 0x200628e1090>
```

[20]:
```
# scores[1] will correspond to accuracy if we pass metrics=['accuracy']
scores = model.evaluate(X_test, y_test, verbose=0)
print('Test accuracy:', scores[1])
```

```
Test accuracy: 0.8744400143623352
```

[ ]:

[ ]: