Sentiment_Analysis_RNN_1

February 21, 2025

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
%cd /content/drive/MyDrive/Deep_Learning_With_Tensorflow
```

```
/content/drive/MyDrive/Deep_Learning_With_Tensorflow
```

[4]:
```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load IMDB dataset (restrict vocabulary to 10,000 most frequent words)
num_words = 10000  # Vocabulary size limit
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_words)

# Pad sequences to a fixed length
max_length = 200
x_train = pad_sequences(x_train, maxlen=max_length)
x_test = pad_sequences(x_test, maxlen=max_length)

# Define vocabulary size correctly
vocab_size = num_words  # Fix vocab size

# Build a neural network using Simple RNN
embedding_dim = 50
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim,␣
 ↪input_shape=(max_length,)),
    SimpleRNN(100, activation='tanh'),  # Added Simple RNN Layer
    Dense(1, activation='sigmoid')  # Output layer for binary classification
])

# Compile the model
```

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

# Display the model summary
model.summary()

# Train the model
model.fit(x_train, y_train, epochs=3, batch_size=32)  # Train for a small␣
 ↪number of epochs for demonstration

# Evaluate Model
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Extract the learned word embeddings
embedding_layer = model.layers[0]
weights = embedding_layer.get_weights()[0]  # Shape: (vocab_size, embedding_dim)

# Load IMDB word index
imdb_word_index = imdb.get_word_index()

# Reconstruct word index as per load_data() conventions
word_index = {word: (index + 3) for word, index in imdb_word_index.items()}  #␣
 ↪Shift indices by 3
word_index["<PAD>"] = 0
word_index["<START>"] = 1
word_index["<UNK>"] = 2
word_index["<UNUSED>"] = 3

# Reverse lookup dictionary for saving embeddings
reverse_word_index = {i: word for word, i in word_index.items()}

print("\nPrinting reverse word index first few items:\n")
print(reverse_word_index.get(0), " ", reverse_word_index.get(1), " ",␣
 ↪reverse_word_index.get(2), "  ", reverse_word_index.get(3))

# Save the learned word embeddings
with open("word_embeddings.txt", "w", encoding="utf-8") as file:
    for i in range(1, vocab_size):  # Skip padding index (0)
        word = reverse_word_index.get(i, "<UNK>")  # Use <UNK> for missing words
        embedding = " ".join(map(str, weights[i]))  # Convert embedding to␣
 ↪space-separated string
        file.write(f"{word} {embedding}\n")

print("Word embeddings saved to word_embeddings.txt")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:93:
```

```
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 200, 50)	500,000
simple_rnn_1 (SimpleRNN)	(None, 100)	15,100
dense_1 (Dense)	(None, 1)	101

Total params: 515,201 (1.97 MB)

Trainable params: 515,201 (1.97 MB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/3
782/782             62s 76ms/step -
accuracy: 0.5512 - loss: 0.6799
Epoch 2/3
782/782             79s 73ms/step -
accuracy: 0.7376 - loss: 0.5255
Epoch 3/3
782/782             58s 74ms/step -
accuracy: 0.7603 - loss: 0.5035
782/782             15s 19ms/step -
accuracy: 0.7603 - loss: 0.5098
Test Accuracy: 76.24%
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/imdb_word_index.json
1641221/1641221             0s
0us/step

Printing reverse word index first few items:
```

```
<PAD>   <START>   <UNK>    <UNUSED>
Word embeddings saved to word_embeddings.txt
```