PMDS603P Deep Learning Lab Experiment 10

October 2025

1 Work to do today

Note: Make a single PDF file of the work you are doing in a Jupyter notebook. Upload with the proper format. Please mention your name and roll no properly with the Experiment number on the first page of your submission.

Today we will look at how we can define a simple RNN model to perform sentiment analysis using imdb movie dataset.

Question 1: First, we will try to load the dataset. Here we Load IMDB dataset with only the top 10,000 most frequent words. Remaining words are discarded in this model. They are assigned some special tokens. Here, each review is stored in the form of a sequence of

```
from tensorflow.keras.layers import SimpleRNN, LSTM, GRU, Dense, Embedding
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
import numpy as np
# Getting reviews with words that come under 10000
# most occurring words in the entire
# corpus of textual review data
vocab_size = 10000
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)
print(x_train.size)
print(x_test.size)
```

integers where each integer corresponds to one word in the vocabulary.

Now try to print and check x_train[0], it would be a sequence of array which corresponds to the first review.

If you want to see the mappings, it is available. You can display the dictionary with the word as the key and the respective word as the value in the form of a dictionary. Next we

```
index = imdb.get_word_index()
print(index)
```

will just reverse the key value pairs of the dictionary and print and check one such review

```
reverse_index = dict([(value, key) for (key, value) in index.items()])
print(reverse_index[1])
decoded = " ".join( [reverse_index.get(i-3, "#") for i in x_test[1]] )
print(y_test[1])
print(decoded)
```

and the sentiment whether its positive or negative review. Here i-3 because the first three are special tokens.

Next find the maximum and minimum length of the reviews

```
print("Max length of a review:: ", len(max((x_train+x_test), key=len)))
print("Min length of a review:: ", len(min((x_train+x_test), key=len)))
```

Next we will try to fix a length of each review say 400 or 200 words. So the ones which have more than that number of words in the review will be trimmed and others will be padded with 0.

```
from tensorflow.keras.preprocessing import sequence

# Keeping a fixed length of all reviews to max 400 words
max_words = 200

x_train = sequence.pad_sequences(x_train, maxlen=max_words)
x_test = sequence.pad_sequences(x_test, maxlen=max_words)
print(x_train[0])
print()

x_valid, y_valid = x_train[:64], y_train[:64]
x_train_, y_train_ = x_train[64:], y_train[64:]
print(x_train_.shape)

# fixing every word's embedding size to be 32
embd_len = 64
```

May be you can modify the x_valid and the y_valid as the first 500 reviews instead of 64 and the remainig as x_train_ and y_train_ Next we fix the length of word embeddings.That is we can decide the dimension of the space onto which we try to map the words on to. The concept of embedding is to represent the words in some Euclidean space with the dimension of which is specified as embd_len. The embedding layer learns the embedding matrix and can be used for proper predictions while using the testing set.

```
RNN_model = Sequential()
RNN_model.add(Embedding(input_dim=vocab_size,
                    output_dim=embd_len, input_length = max_words))
RNN_model.add(SimpleRNN(128,
                        activation='tanh',
                        return_sequences=False))

RNN_model.add(Dense(1, activation='sigmoid'))
```

Next we can compile and fit the model. Let us find the predictions next. Now print the

```
RNN_model.compile(
    loss="binary_crossentropy",
    optimizer='adam',
    metrics=['accuracy']
)

RNN_model.fit(x_train_, y_train_,
                        batch_size=64,
                        epochs=3,
                        verbose=1,
                        validation_data=(x_valid, y_valid))

RNN_model.summary()

print()
print("Simple_RNN Score---> ", RNN_model.evaluate(x_test, y_test, verbose=0))
```

```
y_pred_prob = RNN_model.predict(x_test)
y_pred = (y_pred_prob > 0.5).astype(int).reshape(-1)  # Threshold at 0.5
```

various performance metrics.

Question 2: Next, build an LSTM model with two LSTM layers with an appropriate choice of max_length, embedding length etc and print the results