End to End Deep Learning Project Using Simple RNN

In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
```

In [2]:
```
tf.keras.backend.clear_session()
```

In [3]:
```
import gc
gc.collect()
```

Out[3]:
```
0
```

In [4]:
```
## load the imdb dataset
max_features = 10000  # vocabulary(number of words to consider as features)
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=max_features)

print(f'Training data shape: {X_train.shape}, Training labels shape: {y_train.shape}')
print(f'Testing data shape: {X_test.shape}, Testing labels shape: {y_test.shape}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Training data shape: (25000,), Training labels shape: (25000,)
Testing data shape: (25000,), Testing labels shape: (25000,)
```

In [ ]:
```
## Inspect a sample review
sample_review = X_train[0]
sample_label = y_train[0]

print(f'Sample review (as word indices): {sample_review},\n{sample_label}')
```

```
Sample review (as word indices): [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 10
0, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 4
47, 4, 192, 50, 16, 6, 147, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4, 22,
17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 6
19, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 25
6, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 5
6, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 6
5, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32],
1
```

In [6]:
```
# Mapping of word indices to words
word_index = imdb.get_word_index()
print(f'word_index: {list(word_index.items())[:10]}')  # Print first 10 entries

reverse_word_index = {value: key for (key, value) in word_index.items()}
print(f'reverse_word_index: {list(reverse_word_index.items())[:10]}')  # Print first 10 entries
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1641221/1641221 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
word_index: [('fawn', 34701), ('tsukino', 52006), ('nunnery', 52007), ('sonja', 16816), ('vani', 63951), ('woods', 1408), ('spi
ders', 16115), ('hanging', 2345), ('woody', 2289), ('trawling', 52008)]
reverse_word_index: [(34701, 'fawn'), (52006, 'tsukino'), (52007, 'nunnery'), (16816, 'sonja'), (63951, 'vani'), (1408, 'wood
s'), (16115, 'spiders'), (2345, 'hanging'), (2289, 'woody'), (52008, 'trawling')]
```

In [7]:
```
decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in sample_review])
print(f'Decoded review: {decoded_review}')
```

```
Decoded review: ? this film was just brilliant casting location scenery story direction everyone's really suited the part they
played and you could just imagine being there robert ? is an amazing actor and now the same being director ? father came from t
he same scottish island as myself so i loved the fact there was a real connection with this film the witty remarks throughout t
he film were great it was just brilliant so much that i bought the film as soon as it was released for ? and would recommend it
to everyone to watch and the fly fishing was amazing really cried at the end it was so sad and you know what they say if you cr
y at a film it must have been good and this definitely was also ? to the two little boy's that played the ? of norman and paul
they were just brilliant children are often left out of the ? list i think because the stars that play them all grown up are su
ch a big profile for the whole film but these children are amazing and should be praised for what they have done don't you thin
k the whole story was so lovely because it was true and was someone's life after all that was shared with us all
```

In [8]:
```
# Pad sequences to ensure uniform input length
maxlen = 500  # maximum length of a review

X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
```

In [9]:
```
print(f'Padded training data shape: {X_train.shape}')
```

```
Padded training data shape: (25000, 500)
```

In [10]:
```
## Train a simple RNN model

model = Sequential()
```

```
model.add(Embedding(max_features, 128))  # Embedding layer
model.add(SimpleRNN(128, activation='relu'))  # Simple RNN layer
model.add(Dense(1, activation='sigmoid'))  # Output layer
```

In [11]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [12]:
```
## Create an instance of EarlyStopping Callback
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

In [13]:
```
## Train a simple RNN model
model.fit(
    X_train,
    y_train,
    epochs=10,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping]
)
```

```
Epoch 1/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 29s 41ms/step - accuracy: 0.5610 - loss: 2022386171904.0000 - val_accuracy: 0.5530 - val_loss: 0.6
678
Epoch 2/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.6115 - loss: 0.6687 - val_accuracy: 0.7416 - val_loss: 0.5186
Epoch 3/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.8430 - loss: 0.4632 - val_accuracy: 0.8104 - val_loss: 0.4247
Epoch 4/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.9171 - loss: 0.2745 - val_accuracy: 0.8380 - val_loss: 0.3835
Epoch 5/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.7695 - loss: 49482.2812 - val_accuracy: 0.7402 - val_loss: 0.5610
Epoch 6/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.8430 - loss: 0.3948 - val_accuracy: 0.7732 - val_loss: 0.5359
Epoch 7/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.8911 - loss: 0.3203 - val_accuracy: 0.7862 - val_loss: 0.5222
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 41s 40ms/step - accuracy: 0.9063 - loss: 0.2775 - val_accuracy: 0.7980 - val_loss: 0.5293
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 25s 40ms/step - accuracy: 0.9209 - loss: 0.2416 - val_accuracy: 0.8042 - val_loss: 0.5159
```

Out[13]: <keras.src.callbacks.history.History at 0x7f69756b0530>

In [14]:
```
## Save the model
model.save('simple_rnn_imdb.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.
save_model(model, 'my_model.keras')`.
```