**Exercise: 2 - IMDB Movie Review Sentiment Analysis**

**Files:**

## Project Structure

* **Simplernn.py**: Contains code for training the RNN model.
* **Prediction.py**: Contains code for making predictions with the trained model.
* **main.py**: Streamlit app for user interaction and sentiment analysis.

## 1. Data Preparation

**1.1. Load the IMDB Dataset**

|  |
| --- |
| from tensorflow.keras.datasets import imdb  max\_features = 10000 # Vocabulary size  (X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=max\_features) |

**1.2. Inspect a Sample Review and Label**

|  |
| --- |
| sample\_review = X\_train[0]  sample\_label = y\_train[0]  print(f"Sample review (as integers): {sample\_review}")  print(f'Sample label: {sample\_label}') |

**1.4. Map Indices Back to Words**

|  |
| --- |
| from tensorflow.keras.datasets import imdb  word\_index = imdb.get\_word\_index()  reverse\_word\_index = {value: key for key, value in word\_index.items()}  decoded\_review = ' '.join([reverse\_word\_index.get(i - 3, '?') for i in sample\_review])  print(f'Decoded review: {decoded\_review}') |

**1.5 Preprocess the data**

|  |
| --- |
| from tensorflow.keras.preprocessing import sequence  max\_len = 500  X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_len)  X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_len) |

**2. Model Training**

**2.1. Define Model Architecture**

|  |
| --- |
| from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Embedding, SimpleRNN, Dense  model = Sequential()  model.add(Embedding(max\_features, 128, input\_length=max\_len)) # Embedding Layer  model.add(SimpleRNN(128, activation='relu'))  model.add(Dense(1, activation="sigmoid")) |

**2.2. Compile the Model**

|  |
| --- |
| model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) |

**2.3. Create and Apply Early Stopping**

|  |
| --- |
| from tensorflow.keras.callbacks import EarlyStopping  earlystopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True) |

**2.4. Train the Model**

|  |
| --- |
| history = model.fit(  X\_train, y\_train, epochs=10, batch\_size=32,  validation\_split=0.2,  callbacks=[earlystopping]  ) |

**2.5. Save the Model**

|  |
| --- |
| model.save('simple\_rnn\_imdb\_output.h5') |

**2.6. Save Training History**

|  |
| --- |
| import pandas as pd  history\_df = pd.DataFrame(history.history)  history\_df.to\_csv('training\_history.csv', index=False) |

**3. Prediction**

**3.1. Load the Model**

|  |
| --- |
| from tensorflow.keras.models import load\_model  model = load\_model('simple\_rnn\_imdb\_output.h5') |

**3.2. Define Helper Functions**

* **Decode Review:**

|  |
| --- |
| * def decode\_review(encoded\_review): * return ' '.join([reverse\_word\_index.get(i - 3, '?') for i in encoded\_review]) |

* **Preprocess Text:**

|  |
| --- |
| * def preprocess\_text(text): * words = text.lower().split() * encoded\_review = [word\_index.get(word, 2) + 3 for word in words] * padded\_review = sequence.pad\_sequences([encoded\_review], maxlen=500) * return padded\_review |

* **Predict Sentiment:**

|  |
| --- |
| * def predict\_sentiment(review): * preprocessed\_input = preprocess\_text(review) * prediction = model.predict(preprocessed\_input) * sentiment = 'Positive' if prediction[0][0] > 0.5 else 'Negative' * return sentiment, prediction[0][0] |

**3.3. Example Review for Prediction**

|  |
| --- |
| example\_review = "This movie was fantastic! The acting was great and the plot was thrilling."  sentiment, score = predict\_sentiment(example\_review)  print(f'Review: {example\_review}')  print(f'Sentiment: {sentiment}')  print(f'Prediction Score: {score}') |

## 4. Streamlit Application

**4.1. Set Up Streamlit App**

|  |
| --- |
| import streamlit as st  st.title('IMDB Movie Review Sentiment Analysis')  st.write('Enter a movie review to classify it as positive or negative.') |

**4.2. User Input and Prediction**

|  |
| --- |
| user\_input = st.text\_area('Movie Review')  if st.button('Classify'):  preprocessed\_input = preprocess\_text(user\_input)  prediction = model.predict(preprocessed\_input)  sentiment = 'Positive' if prediction[0][0] > 0.5 else 'Negative'  st.write(f'Sentiment: {sentiment}')  st.write(f'Prediction Score: {prediction[0][0]}')  else:  st.write('Please enter a movie review.') |

**4.3. Run Streamlit App**

|  |
| --- |
| streamlit run main.py |

**Execution Links:**

|  |
| --- |
|  **Network link:** http://192.168.0.199:8501/   **Localhost:** http://localhost:8501/ |