In [ ]:
```
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 load_model
```

In [2]:
```
word_index = imdb.get_word_index()

reverse_word_index = {value: key for (key, value) in word_index.items()}
```

In [3]:
```
# Load the pre-trained model with Relu activation
model = load_model('simple_rnn_imdb.h5')
model.summary()
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(32, 500, 128)	1,280,000
simple_rnn (SimpleRNN)	(32, 128)	32,896
dense (Dense)	(32, 1)	129

Total params: 1,313,027 (5.01 MB)

Trainable params: 1,313,025 (5.01 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 2 (12.00 B)

In [7]:
```
import re
```

In [8]:
```
# Helper function to decode reviews
def decode_review(encoded_review):
    return ' '.join([reverse_word_index.get(i - 3, '?') for i in encoded_review])

# Function to preprocess user input
def preprocess_review(review, maxlen=500):
    # Simple preprocessing: lowercase and split by spaces
    # Better preprocessing: handle punctuation and lowercase
    # Remove punctuation and convert to lowercase
    review = re.sub(r'[^a-zA-Z\s]', '', review.lower())
    words = review.split()
    encoded_review = []
    for word in words:
        index = word_index.get(word, 2)  # 2 is the index for 'unknown' words
        encoded_review.append(index + 3)  # Offset by 3 for special tokens
    padded_review = sequence.pad_sequences([encoded_review], maxlen=maxlen)
    return padded_review
```

```
  for word in words:
        index = word_index.get(word, 2)  # 2 is the index for 'unknown' words
        encoded_review.append(index + 3)  # Offset by 3 for special tokens
```

- Each word is converted to its corresponding integer index using the IMDB word index dictionary
- If a word isn't found in the dictionary, it gets index 2 (unknown word token)
- Adds 3 to each index to account for reserved special tokens:
  - Index 0: Padding
  - Index 1: Start of sequence
  - Index 2: Unknown word
  - Index 3+: Actual vocabulary words

In [9]:
```
## Prediction function

def predict_review(review):

    preprocessed_review = preprocess_review(review)

    prediction = model.predict(preprocessed_review)

    sentiment = 'Positive' if prediction[0][0] >= 0.5 else 'Negative'

    return sentiment, prediction[0][0]
```

In [10]:
```
# User Input and Prediction
# Example review
# example_review = "The movie was fantastic! I really loved it and would watch it again."
example_review = "The movie was fantastic! The acting was great and the plot was thrilling."
```

```
sentiment, confidence = predict_review(example_review)

print(f'Review: {example_review}')
print(f'Sentiment: {sentiment} (Confidence: {confidence:.4f})')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
Review: The movie was fantastic! The acting was great and the plot was thrilling.
Sentiment: Positive (Confidence: 0.6722)
```

In [11]:
```
# Let's create the corrected prediction functions
def predict_review_corrected(review):
    """Updated prediction function with improved preprocessing"""
    preprocessed_review = preprocess_review(review)
    prediction = model.predict(preprocessed_review, verbose=0)
    sentiment = 'Positive' if prediction[0][0] >= 0.5 else 'Negative'
    return sentiment, prediction[0][0]

# Test the corrected function
print("=== CORRECTED PREDICTION FUNCTION ===")
test_reviews = [
    "The movie was fantastic! The acting was great and the plot was thrilling.",
    "This movie is terrible and boring.",
    "Amazing film with great acting!",
    "I loved every minute of it.",
    "Worst movie ever made.",
    "Absolutely brilliant and entertaining."
]

for review in test_reviews:
    sentiment, confidence = predict_review_corrected(review)
    print(f"Review: {review}")
    print(f"Prediction: {sentiment} (Confidence: {confidence:.4f})")
    print()
```

```
=== CORRECTED PREDICTION FUNCTION ===
Review: The movie was fantastic! The acting was great and the plot was thrilling.
Prediction: Positive (Confidence: 0.6722)

Review: This movie is terrible and boring.
Prediction: Negative (Confidence: 0.0110)

Review: Amazing film with great acting!
Prediction: Positive (Confidence: 0.7480)

Review: I loved every minute of it.
Prediction: Negative (Confidence: 0.1817)

Review: Worst movie ever made.
Prediction: Negative (Confidence: 0.2046)

Review: Absolutely brilliant and entertaining.
Prediction: Positive (Confidence: 0.6146)
```

In [ ]: