```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tabulate import tabulate
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
# Load dataset (Replace with actual dataset loading)
data = pd.read_json("train.jsonl", lines=True)
```

```
# Extract relevant columns
X = data['postText'].apply(lambda x: x[0])  # Extract text from list
y = data['tags'].apply(lambda x: 1 if 'passage' in x else 0)  # Binary classification (passage vs. other)
```

```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Tokenization & Padding
MAX_VOCAB_SIZE = 10000
MAX_SEQUENCE_LENGTH = 100

tokenizer = Tokenizer(num_words=MAX_VOCAB_SIZE, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)
```

```
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(X_train_seq, maxlen=MAX_SEQUENCE_LENGTH, padding='post', truncating='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=MAX_SEQUENCE_LENGTH, padding='post', truncating='post')

# Define Model Configurations
model_configs = {
    "LSTM Model 1": {"units": 64, "dropout": 0.2, "bidirectional": False},
    "LSTM Model 2": {"units": 128, "dropout": 0.4, "bidirectional": False},
    "LSTM Model 3": {"units": 64, "dropout": 0.2, "bidirectional": True}
}
```

```
# Convert model configurations to DataFrame
config_df = pd.DataFrame.from_dict(model_configs, orient='index')
print("\n 🔹 LSTM Model Configurations:")
print(tabulate(config_df, headers='keys', tablefmt='grid'))
```

```
 🔹 LSTM Model Configurations:
+--------------+---------+-----------+-----------------+
|              |   units |   dropout | bidirectional   |
+==============+=========+===========+=================+
| LSTM Model 1 |      64 |       0.2 | False           |
+--------------+---------+-----------+-----------------+
| LSTM Model 2 |     128 |       0.4 | False           |
+--------------+---------+-----------+-----------------+
| LSTM Model 3 |      64 |       0.2 | True            |
+--------------+---------+-----------+-----------------+
```

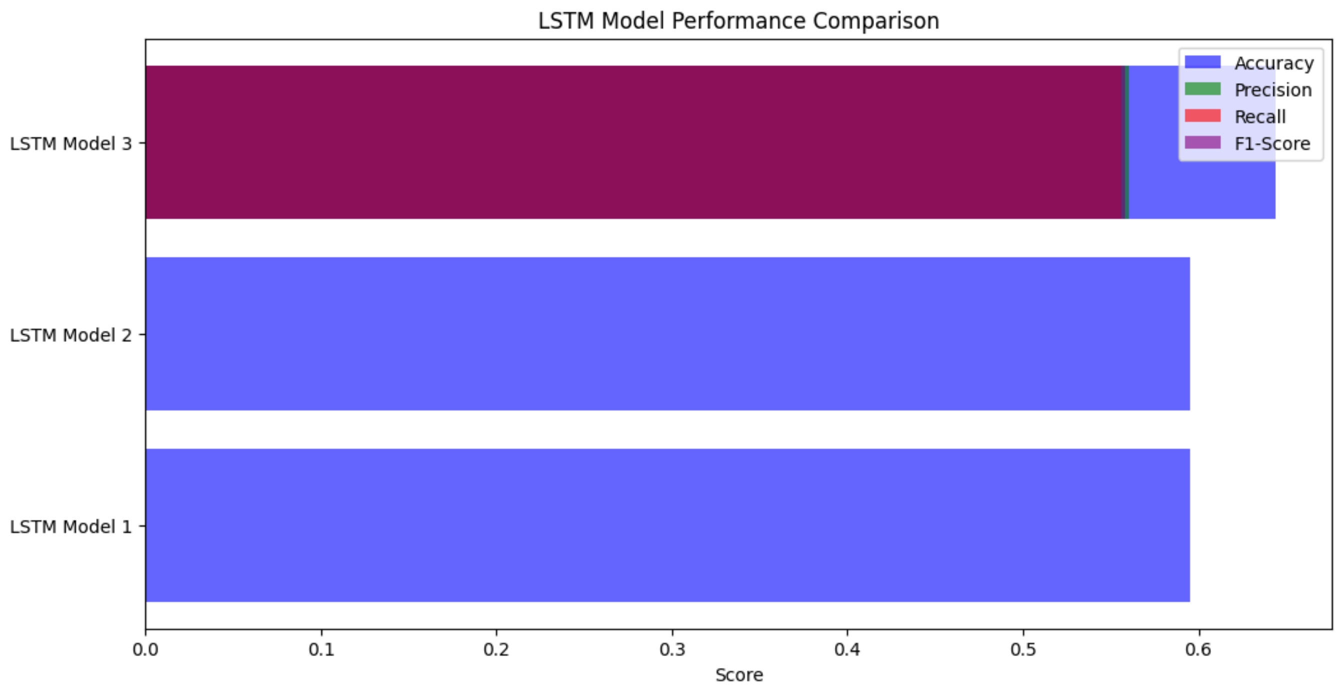
```
# Dictionary to Store Results
results = []

# Train and Evaluate Different LSTM Models
for model_name, config in model_configs.items():
    model = Sequential()
    model.add(Embedding(MAX_VOCAB_SIZE, 128, input_length=MAX_SEQUENCE_LENGTH))

    if config["bidirectional"]:
        model.add(Bidirectional(LSTM(config["units"], return_sequences=False)))
    else:
        model.add(LSTM(config["units"], return_sequences=False))

    model.add(Dropout(config["dropout"]))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
    print(f"\nTraining {model_name}...")
    history = model.fit(X_train_pad, y_train, epochs=5, batch_size=32, validation_data=(X_test_pad, y_test), verbose=1)

    # Predictions & Evaluation
    y_pred = (model.predict(X_test_pad) > 0.5).astype(int)

    results.append([
        model_name,
        accuracy_score(y_test, y_pred),
        precision_score(y_test, y_pred),
        recall_score(y_test, y_pred),
        f1_score(y_test, y_pred)
    ])
```

```
Training LSTM Model 1...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated.
  warnings.warn(
Epoch 1/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 103ms/step - accuracy: 0.5900 - loss: 0.6743 - val_accuracy: 0.5953 - val_loss: 0.6760
Epoch 2/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 7s 67ms/step - accuracy: 0.6091 - loss: 0.6737 - val_accuracy: 0.5953 - val_loss: 0.6749
Epoch 3/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 70ms/step - accuracy: 0.5933 - loss: 0.6777 - val_accuracy: 0.5953 - val_loss: 0.6758
Epoch 4/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 6s 80ms/step - accuracy: 0.6039 - loss: 0.6726 - val_accuracy: 0.5953 - val_loss: 0.6752
Epoch 5/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - accuracy: 0.5807 - loss: 0.6813 - val_accuracy: 0.5953 - val_loss: 0.6755
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step

Training LSTM Model 2...
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated.
  warnings.warn(
80/80 ━━━━━━━━━━━━━━━━━━━━ 20s 206ms/step - accuracy: 0.5948 - loss: 0.6816 - val_accuracy: 0.5953 - val_loss: 0.6812
Epoch 2/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 14s 177ms/step - accuracy: 0.6185 - loss: 0.6746 - val_accuracy: 0.5953 - val_loss: 0.6757
Epoch 3/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 20s 174ms/step - accuracy: 0.5920 - loss: 0.6783 - val_accuracy: 0.5953 - val_loss: 0.6758
Epoch 4/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 21s 176ms/step - accuracy: 0.6096 - loss: 0.6707 - val_accuracy: 0.5953 - val_loss: 0.6776
Epoch 5/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 15s 192ms/step - accuracy: 0.5983 - loss: 0.6758 - val_accuracy: 0.5953 - val_loss: 0.6749
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step

Training LSTM Model 3...
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated.
  warnings.warn(
80/80 ━━━━━━━━━━━━━━━━━━━━ 14s 133ms/step - accuracy: 0.5938 - loss: 0.6705 - val_accuracy: 0.6453 - val_loss: 0.6222
Epoch 2/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 126ms/step - accuracy: 0.7851 - loss: 0.4854 - val_accuracy: 0.6625 - val_loss: 0.6214
Epoch 3/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 9s 112ms/step - accuracy: 0.9308 - loss: 0.1960 - val_accuracy: 0.6469 - val_loss: 0.8685
Epoch 4/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 11s 120ms/step - accuracy: 0.9825 - loss: 0.0669 - val_accuracy: 0.6547 - val_loss: 1.1168
Epoch 5/5
80/80 ━━━━━━━━━━━━━━━━━━━━ 11s 126ms/step - accuracy: 0.9977 - loss: 0.0188 - val_accuracy: 0.6438 - val_loss: 1.4250
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step
```

```
# Convert results to DataFrame
results_df = pd.DataFrame(results, columns=["Model", "Accuracy", "Precision", "Recall", "F1-Score"])
print("\nLSTM Model Performance Comparison:")
print(results_df)
```

```
LSTM Model Performance Comparison:
          Model  Accuracy  Precision    Recall  F1-Score
0  LSTM Model 1  0.595313   0.000000  0.000000   0.00000
1  LSTM Model 2  0.595313   0.000000  0.000000   0.00000
2  LSTM Model 3  0.643750   0.560311  0.555985   0.55814
```

```
# Visualization
plt.figure(figsize=(12, 6))
metrics = ["Accuracy", "Precision", "Recall", "F1-Score"]
colors = ["blue", "green", "red", "purple"]

for i, metric in enumerate(metrics):
    plt.barh(results_df["Model"], results_df[metric], color=colors[i], alpha=0.6, label=metric)
plt.xlabel("Score")
plt.title("LSTM Model Performance Comparison")
plt.legend()
plt.show()
```



```
Start coding or generate with AI.
```