In [1]:

```
!pip install Keras-Preprocessing
```

```
Collecting Keras-Preprocessing
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.6/42.6 kB 1.8 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/lib/python3.7/site-packages
(from Keras-Preprocessing) (1.21.6)
Requirement already satisfied: six>=1.9.0 in /opt/conda/lib/python3.7/site-packages
(from Keras-Preprocessing) (1.16.0)
Installing collected packages: Keras-Preprocessing
Successfully installed Keras-Preprocessing-1.1.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment instead: https://pip.pypa.io/warnings/venv
```

In [2]:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Embedding, LSTM, Dropout, Dense
from tensorflow.keras.models import Model
import numpy as np
import pandas as pd
import os
import re

import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from keras.preprocessing.text import Tokenizer

from keras.layers import Input, Dense, Dropout, Embedding, LSTM, Conv1D, MaxPooling1D,
from keras.models import Model
from keras.optimizers import Adam
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
```

In [3]:

```
from sklearn.model_selection import train_test_split

from keras.layers import Input, Embedding, LSTM, Dropout, Dense
from keras.callbacks import EarlyStopping
```

In [5]:

```
df = pd.read_csv('/kaggle/input/answerscript4/p3.csv')
```

In [6]:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[6]:

```
True
```

In [7]:

```
df_ind = df.loc[df['essay_set']==3]

df_ind = df_ind[['essay','domain1_score']]

df_ind = df_ind.sort_values(by= ["domain1_score"], ascending=False)
answer_sheet = df_ind.iloc[0]['essay']
df_ind = df_ind.drop(0)
```

```
students_answers = list(df['essay'].values)
marks_org = list(df['domain1_score'].values)
t=max(marks_org)
#df.apply(preprocess_text1,pandas column name)
```

In [8]:

```
df.head()
```

Out[8]:

	Unnamed: 0	essay_id	essay_set	essay	rater1_domain1	rater2_domain1	rater3_domain1	doma
0	3583	5978	3	The features of the setting affect the cyclist...	1.0	1.0	NaN	
1	3584	5979	3	The features of the setting affected the cycli...	2.0	2.0	NaN	
2	3585	5980	3	Everyone travels to unfamiliar places. Sometim...	1.0	1.0	NaN	
3	3586	5981	3	I believe the features of the cyclist affected...	1.0	1.0	NaN	
4	3587	5982	3	The setting effects the cyclist because of the...	2.0	2.0	NaN	

5 rows × 29 columns

In [9]:

```
# Set the random seed for reproducibility
np.random.seed(42)

# Set the maximum sequence length and embedding dimension
MAX_SEQUENCE_LENGTH = 1000
EMBEDDING_DIM = 100

# Set the number of LSTM units and dropout rate
NUM_LSTM_UNITS = 128
DROPOUT_RATE = 0.2

# Set the batch size and number of epochs
```

```
BATCH_SIZE = 64
EPOCHS = 10

# Define the function to preprocess the text
def preprocess_text(x, remove_stopwords=False):
    x = x.lower()
    x = re.sub("[^a-z\s+]","",x)
    if remove_stopwords:
        x = " ".join([word for word in x.split() if word not in stopwords.words('engli
    return x

# Load the essays dataset
essays_df = pd.read_csv('/kaggle/input/answerscript4/p7.csv', encoding='latin-1')

# Remove essays that have a domain1_score of NaN
essays_df = essays_df[~essays_df['domain1_score'].isna()]

# Remove stopwords from the essays
essays_df['essay'] = essays_df['essay'].apply(preprocess_text, remove_stopwords=True)
essays_df['expected']=answer_sheet
```

In [10]:

```
# Split the dataset into training, validation, and test sets
train_df, test_df = train_test_split(essays_df, test_size=0.2, random_state=42)
train_df, val_df = train_test_split(train_df, test_size=0.2, random_state=42)

# Tokenize the texts
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_df['essay'])

# Convert the texts to sequences and pad them to the specified maximum length
X_expected_train = pad_sequences(tokenizer.texts_to_sequences(train_df['expected']), m
X_actual_train = pad_sequences(tokenizer.texts_to_sequences(train_df['essay']), maxlen
y_train = train_df['domain1_score'].values /t

X_expected_val = pad_sequences(tokenizer.texts_to_sequences(val_df['expected']), maxle
X_actual_val = pad_sequences(tokenizer.texts_to_sequences(val_df['essay']), maxlen=MAX
y_val = val_df['domain1_score'].values /t

X_expected_test = pad_sequences(tokenizer.texts_to_sequences(test_df['expected']), max
X_actual_test = pad_sequences(tokenizer.texts_to_sequences(test_df['essay']), maxlen=M
y_test = test_df['domain1_score'].values /t

# Define the inputs and embedding layer
expected_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32', name='expected_inp
actual_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32', name='actual_input')
embedding_layer = Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=EMBEDD

# Encode the inputs with the embedding layer
expected_encoded = embedding_layer(expected_input)
actual_encoded = embedding_layer(actual_input)
```

In [11]:

```
# Define the LSTM layer
lstm_layer = LSTM(NUM_LSTM_UNITS)

# Define the dropout layer
dropout_layer = Dropout(DROPOUT_RATE)

# Define the output layer
output_layer = Dense(1, activation='sigmoid')
```

```
# Encode the expected and actual inputs
expected_encoded = embedding_layer(expected_input)
actual_encoded = embedding_layer(actual_input)

# Pass the expected and actual inputs through the LSTM layer
expected_output = lstm_layer(expected_encoded)
actual_output = lstm_layer(actual_encoded)

# Apply dropout to the LSTM outputs
expected_output = dropout_layer(expected_output)
actual_output = dropout_layer(actual_output)

# Pass the LSTM outputs through the output layer
expected_output = output_layer(expected_output)
actual_output = output_layer(actual_output)


model_inputs = [expected_input, actual_input]
model_outputs = [actual_output]

# Create the model
model = Model(inputs=model_inputs, outputs=model_outputs)

# Compile the model with MAE as the loss function
model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

# Train the model
model.fit(x=[X_expected_train, X_actual_train], y=y_train, batch_size=BATCH_SIZE, epoc

# Evaluate the model
loss, mae = model.evaluate([X_expected_test, X_actual_test], y_test, batch_size=BATCH_
print('Test Loss:', loss)
print('Test MAE:', mae)
```

```
Epoch 1/10
20/20 [==============================] - 31s 1s/step - loss: 4.5843 - mae: 4.5843 - v
al_loss: 4.2701 - val_mae: 4.2701
Epoch 2/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 3/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 4/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 5/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 6/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 7/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 8/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 9/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
Epoch 10/10
20/20 [==============================] - 28s 1s/step - loss: 4.3746 - mae: 4.3746 - v
al_loss: 4.2700 - val_mae: 4.2700
3/3 [==============================] - 1s 434ms/step - loss: 4.2696 - mae: 4.2696
Test Loss: 4.26964807510376
Test MAE: 4.26964807510376
```

```
In [ ]:
```