Betterzila (Changumangu Solutions LLP)

Generative AI internship

(Assignment)

Objective:

Develop a simple transformer-based model to solve a specific problem, such as text classification, sentiment analysis, or language translation.

Dataset used :

I have used the “IMDB reviews” dataset from kaggle for the task of sentiment analysis.

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

The dataset has 50,000 rows.

Data Pre-processing:

For data preprocessing, I begin by loading the dataset from the provided CSV file. I transform the 'sentiment' column into numerical values, mapping 'positive' to 1 and 'negative' to 0.

```
[44] df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})
```

```
[45] df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

To ensure proper training, I split the data into training, validation, and test sets using the train_test_split function.

```
[17] train_data, test_data = train_test_split(df, test_size=0.2, random_state=42)
     train_data, valid_data = train_test_split(train_data, test_size=0.1, random_state=42)
```

Next, I tokenize and pad the text sequences using TensorFlow's Tokenizer and pad_sequences functions, ensuring consistent sequence lengths for input.

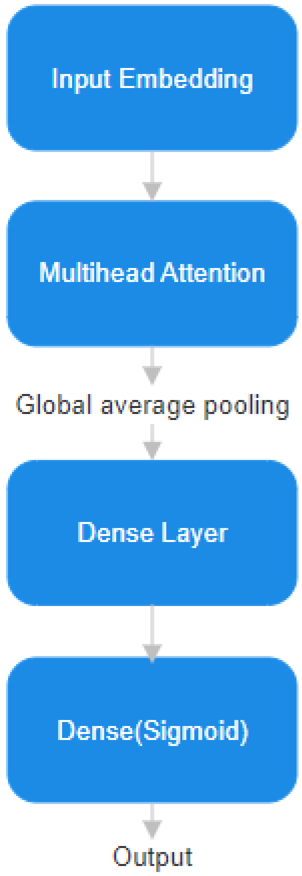
```
[18] max_words = 10000
     max_len = 200

     tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
     tokenizer.fit_on_texts(train_data['review'])
```

```
X_train = pad_sequences(tokenizer.texts_to_sequences(train_data['review']), maxlen=max_len)
X_valid = pad_sequences(tokenizer.texts_to_sequences(valid_data['review']), maxlen=max_len)
X_test = pad_sequences(tokenizer.texts_to_sequences(test_data['review']), maxlen=max_len)
```

<u>Model Architecture:</u>

When creating the model structure, I use TensorFlow's Keras API.
Embedding Layer: This layer helps the model understand the meaning of words.
MultiHeadAttention Layer: This layer looks at different parts of the text at the same time to capture context.
GlobalAveragePooling1D Layer: This layer combines all the information gathered to create a summary of the text.
Dense Layers: These are simple layers that do additional processing.
Sigmoid Activation: This is like a switch that helps the model make a decision - it's set up for binary choices like positive or negative.

```
def transformer_model(input_dim, embed_dim, num_heads, ff_dim, output_dim, max_len):
    inputs = Input(shape=(max_len,))
    x = Embedding(input_dim, embed_dim)(inputs)
    x = MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)(x, x)
    x = GlobalAveragePooling1D()(x)
    x = Dense(ff_dim, activation='relu')(x)
    outputs = Dense(output_dim, activation='sigmoid')(x)
    return Model(inputs=inputs, outputs=outputs)
```



Defining the hyperparameters for transformer model:

```
[21] embed_dim = 64
     num_heads = 4
     ff_dim = 64
     output_dim = 1
```

Initializing the model and starting the training process:

```
[23] model.fit(X_train, train_data['sentiment'], epochs=10, batch_size=64, validation_data=(X_valid, valid_data['sentiment']))
```

Model Evaluation:

I evaluated the model on the test set to check its generalization performance. I compute various metrics, including accuracy, precision, recall, F1-score, and a confusion matrix.

Accuracy provides an overall measure of correct predictions, while precision and recall offer insights into the model's ability to identify positive instances accurately and capture all positive instances, respectively.
F1-score provides a balanced measure between precision and recall.

The confusion matrix breaks down true positive, true negative, false positive, and false negative predictions in detail.

```
y_pred = model.predict(X_test)
y_pred_binary = [1 if pred > 0.5 else 0 for pred in y_pred]
test_acc = accuracy_score(test_data['sentiment'], y_pred_binary)
print(f'Test Accuracy: {test_acc*100:.2f}%')

precision = precision_score(test_data['sentiment'], y_pred_binary)
recall = recall_score(test_data['sentiment'], y_pred_binary)
f1 = f1_score(test_data['sentiment'], y_pred_binary)

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-score: {f1:.4f}')

conf_matrix = confusion_matrix(test_data['sentiment'], y_pred_binary)
print('Confusion Matrix:')
print(conf_matrix)
```

```
313/313 [==============================] - 2s 6ms/step
Test Accuracy: 85.50%
Precision: 0.8446
Recall: 0.8728
F1-score: 0.8585
Confusion Matrix:
[[4152  809]
 [ 641 4398]]
```

The model has an overall accuracy of 86%.

Testing the model on some reviews:

```
new_data = ["Phil the Alien is one of those quirky films where the humour is based around the oddness of everything rather than actual punchl:
sequences = pad_sequences(tokenizer.texts_to_sequences(new_data), maxlen=max_len)
predictions = model.predict(sequences)

predicted_labels = ["positive" if pred > 0.6 else "negative" for pred in predictions]

print(predicted_labels)
```

```
1/1 [==============================] - 0s 24ms/step
['negative']
```

```
[19] new_data = ["Some films just simply should not be remade. This is one of them.In and of itself it is not a bad film. But it fails to capture
sequences = pad_sequences(tokenizer.texts_to_sequences(new_data), maxlen=max_len)
predictions = model.predict(sequences)

predicted_labels = ["positive" if pred > 0.6 else "negative" for pred in predictions]

print(predicted_labels)
```

```
1/1 [==============================] - 0s 24ms/step
['positive']
```

```python
new_data = ["If you want to know the real story of the Wendigo, I suggest you pick up a copy of Algernon Blackwell's original story. This movi
sequences = pad_sequences(tokenizer.texts_to_sequences(new_data), maxlen=max_len)
predictions = model.predict(sequences)

predicted_labels = ["positive" if pred > 0.6 else "negative" for pred in predictions]

print(predicted_labels)
```

```
1/1 [==============================] - 0s 23ms/step
['negative']
```

```python
[21] new_data = ["Tears of Kali is an original yet flawed horror film that delves into the doings of a cult group in India comprised of German psyc
sequences = pad_sequences(tokenizer.texts_to_sequences(new_data), maxlen=max_len)
predictions = model.predict(sequences)

predicted_labels = ["positive" if pred > 0.6 else "negative" for pred in predictions]

print(predicted_labels)
```

```
1/1 [==============================] - 0s 27ms/step
['positive']
```