Annapoornima S

225229101

PDL Lab16. Design of LSTM and GRU RNN for classification of IMDB reviews

In [1]:

```
import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
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, Bidirectional, GRU
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import load_model
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
data = pd.read_csv('imdb.csv', encoding='latin1')
data.head()
```

Out[3]:

	text	label
0	It's been about 14 years since Sharon Stone aw...	0
1	someone needed to make a car payment... this i...	0
2	The Guidelines state that a comment must conta...	0
3	This movie is a muddled mish-mash of clichés f...	0
4	Before Stan Laurel became the smaller half of ...	0

In [5]:

```
english_sto = set(stopwords.words('english'))
```

In [6]:

```
def load_dataset():
    df = pd.read_csv('imdb.csv', encoding='latin1')
    x_data = df['text']
    y_data = df['label']
    x_data = x_data.replace({'[’A-Za-z]' : ' '}, regex = True)
    x_data = x_data.apply(lambda review: [w for w in review.split() if w not in english_sto])
    x_data = x_data.apply(lambda review: [w.lower() for w in review])
    y_data = y_data.replace('positive', 1)
    y_data = y_data.replace('negative', 0)
    return x_data, y_data
x_data, y_data = load_dataset()
print('Reviews')
print(x_data, '\n')
print('Sentiment')
print(y_data)
```

```
Reviews
0     [', 14, -, ', ., ,, ,, ', ., ", 2", -, ", ", ....
1     [..., ..., ', ..., ..., ..., ., ?, ., ., $5.99...
2                                      [., ., ,, ., .]
3     [-, é, ., ,, ,, ., ', ., ,, (, ,, ,, ).<, /><,...
4     [-, ,, ,, ., ,, ', (, ), .<, /><, />, ', ,, ',...
                            ...
81    [., ', ..., ., ', -, -, -, ., (, ).<, /><, />,...
82    [,, 6, .<, /><, />, ', 2, ,, 11, 2001., ,, (, ...
83    [., -, -, ., ,, ,, ., ., ., -, -, ,, ', ,, ,, ...
84    [., ,, ,, ', ., ,, ", ", ., ., ', ., ,, ', /, ...
85    [,, ', ", /, ", ,, ., (, ', ), ", "., (, ), .,...
Name: text, Length: 86, dtype: object

Sentiment
0     0
1     0
2     0
3     0
4     0
     ..
81    0
82    1
83    1
84    0
85    0
Name: label, Length: 86, dtype: int64
```

In [7]:

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size = 0.4)
print('Train Set')
print(x_train, '\n')
print(x_test, '\n')
print('Test Set')
print(y_train, '\n')
print(y_test)
```

```
61    [;, ,, ..., ,..., ', ,, ,, ", ", ,, ,, ..., ., ...
26    [,, ., ,, ', ., ,, ,, ,, ,, (, ), ., ,, .<, />...
27    [", ", .., ., ', ., ', ', !!, ..<, /><, />, 10...
63    [., ,, 5-7, ., ', ,, .<, /><, />, ., ,, ,, ., ...
64    [,, ?, ', ?, ', ?, ., ', ', ..., ?, ?, ?, ?, -...
84    [., ,, ,, ', ., ,, ", ", ., ., ', ., ,, ', /, ...
51                                [., ,, ., ,, ,, ., ., .]
17    [,, ,, ,, ., .<, /><, />, -, ..., ,, ., -, ,, ...
25    [', ., ', ., ., ,, ,, ., ', ., ', ,, ., ., ,, ...
79                   [,, ., ., ., ., ., ,, ,, ., ', ', .]
78    [-8,, (, ), ., 2, (, ), .<, /><, />, -, ?, ,, ...
15    [-, ,, ", ", 100., ,, ,, (, ),, ., -, ', ,, .,...
77    [,, ', -, ', ., ,, ,, ,, ', ', ., ., ,, ', ', ...
76    [5, 6, ., .<, /><, />, ., ', !!!, ., ", "., !,...
24    [,, ,, ,, ., ', ., (, )., ., ., ,, -, ., ', ',...
82    [,, 6, .<, /><, />, ', 2, ,, 11, 2001., ,, (, ...
16    [', ,, ., ", ",, ", ", ,, ,, ., ", ", ,, ., ,,...
3     [-, é, ., ,, ,, ., ', ., ,, (, ,, ,, ).<, /><,...
20    [(, ), 80', ,, ., ,, ., ,, ., ,, ,, ., ", -, "...
```

In [8]:

```
def get_max_length():
    review_length = []
    for review in x_train:
        review_length.append(len(review))
    return int(np.ceil(np.mean(review_length)))
```

In [9]:

```
token   =   Tokenizer(lower=False)
token.fit_on_texts(x_train)
x_train = token.texts_to_sequences(x_train)
x_test = token.texts_to_sequences(x_test)
max_length = get_max_length()
x_train=pad_sequences(x_train, maxlen=max_length, padding='post', truncating='post')
x_test=pad_sequences(x_test, maxlen=max_length, padding='post', truncating='post')

total_words = len(token.word_index) + 1

print(' Encoded X Train\n', x_train, '\n')
print(' Encoded X Test\n', x_test, '\n')
print('Maximum review length: ', max_length)
```

```
 Encoded X Train
 [[  1   1   2 ...   3   2   0]
 [  1   2   3 ...  23   6   7]
 [ 18   1  11 ...  41  41  11]
 ...
 [  1   1   3 ...   9   3   5]
 [  1   1  32 ...   1   1   1]
 [  2 116   2 ...   0   0   0]]

 Encoded X Test
 [[ 3  5  3 ...  1  1  2]
 [ 3  1  8 ...  1  1  3]
 [ 2  3 11 ...  1 12  2]
 ...
 [ 1  2  3 ...  2  4  4]
 [11 11  3 ...  0  0  0]
 [ 2  5  5 ...  0  0  0]]

Maximum review length:  45
```

In [10]:

```
EMBED_DIM = 32
LSTM_OUT = 64

# Assuming x_data is a list of lists, where each inner list contains words
sentences = [' '.join(sentence) for sentence in x_data]

# Calculate the maximum sequence length
max_sequence_length = max(len(sentence.split()) for sentence in sentences)

# Set your_input_length to the maximum sequence length
your_input_length = max_sequence_length

model = Sequential()
model.add(Embedding(total_words, EMBED_DIM, input_length=your_input_length))
model.add(LSTM(LSTM_OUT))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 156, 32)           3776      
                                                                 
 lstm (LSTM)                 (None, 64)                24832     
                                                                 
 dense (Dense)               (None, 32)                2080      
                                                                 
 dense_1 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 30721 (120.00 KB)
Trainable params: 30721 (120.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

In [12]:

```
eckpoint = ModelCheckpoint('model_weights.h5', save_best_only=True, save_weights_only=True, monitor='val_loss', mode='min', verbose=1)
```

In [14]:

```python
modell = Sequential()
modell.add(Embedding(total_words, 32, input_length=max_length))
modell.add(LSTM(32))
modell.add(Dense(32, activation='relu'))
modell.add(Dense(1, activation='sigmoid'))
modell.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
modell.fit(x_train, y_train, batch_size = 128, epochs = 10, callbacks=[checkpoint])
modell.evaluate(x_test, y_test)
print(modell.summary())
```

```
Epoch 1/10
1/1 [==============================] - ETA: 0s - loss: 0.6929 - accuracy: 0.5098WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 5s 5s/step - loss: 0.6929 - accuracy: 0.5098
Epoch 2/10
1/1 [==============================] - ETA: 0s - loss: 0.6921 - accuracy: 0.5686WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 41ms/step - loss: 0.6921 - accuracy: 0.5686
Epoch 3/10
1/1 [==============================] - ETA: 0s - loss: 0.6913 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 32ms/step - loss: 0.6913 - accuracy: 0.5490
Epoch 4/10
1/1 [==============================] - ETA: 0s - loss: 0.6907 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 32ms/step - loss: 0.6907 - accuracy: 0.5490
Epoch 5/10
1/1 [==============================] - ETA: 0s - loss: 0.6900 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 35ms/step - loss: 0.6900 - accuracy: 0.5490
Epoch 6/10
1/1 [==============================] - ETA: 0s - loss: 0.6894 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 31ms/step - loss: 0.6894 - accuracy: 0.5490
Epoch 7/10
1/1 [==============================] - ETA: 0s - loss: 0.6887 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 35ms/step - loss: 0.6887 - accuracy: 0.5490
Epoch 8/10
1/1 [==============================] - ETA: 0s - loss: 0.6881 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 34ms/step - loss: 0.6881 - accuracy: 0.5490
Epoch 9/10
1/1 [==============================] - ETA: 0s - loss: 0.6875 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 33ms/step - loss: 0.6875 - accuracy: 0.5490
Epoch 10/10
1/1 [==============================] - ETA: 0s - loss: 0.6869 - accuracy: 0.5490WARNING:tensorflow:Can save best model on
ly with val_loss available, skipping.
1/1 [==============================] - 0s 35ms/step - loss: 0.6869 - accuracy: 0.5490
2/2 [==============================] - 1s 13ms/step - loss: 0.6999 - accuracy: 0.5429
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_3 (Embedding)     (None, 45, 32)            3776      
                                                                 
 lstm_3 (LSTM)               (None, 32)                8320      
                                                                 
 dense_6 (Dense)             (None, 32)                1056      
                                                                 
 dense_7 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 13185 (51.50 KB)
Trainable params: 13185 (51.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

In [15]:

```
model2 = Sequential()
model2.add(Embedding(total_words, 32, input_length=max_length))
model2.add(Bidirectional(LSTM(32)))
model2.add(Dense(32, activation='relu'))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
model2.fit(x_train, y_train, batch_size = 128, epochs = 10)
model2.evaluate(x_test, y_test)
print(model2.summary())
```

```
Epoch 1/10
1/1 [==============================] - 7s 7s/step - loss: 0.6933 - accuracy: 0.4706
Epoch 2/10
1/1 [==============================] - 0s 106ms/step - loss: 0.6920 - accuracy: 0.5490
Epoch 3/10
1/1 [==============================] - 0s 74ms/step - loss: 0.6910 - accuracy: 0.5490
Epoch 4/10
1/1 [==============================] - 0s 41ms/step - loss: 0.6901 - accuracy: 0.5490
Epoch 5/10
1/1 [==============================] - 0s 46ms/step - loss: 0.6893 - accuracy: 0.5490
Epoch 6/10
1/1 [==============================] - 0s 43ms/step - loss: 0.6885 - accuracy: 0.5490
Epoch 7/10
1/1 [==============================] - 0s 47ms/step - loss: 0.6877 - accuracy: 0.5490
Epoch 8/10
1/1 [==============================] - 0s 86ms/step - loss: 0.6869 - accuracy: 0.5490
Epoch 9/10
1/1 [==============================] - 0s 65ms/step - loss: 0.6861 - accuracy: 0.5490
Epoch 10/10
1/1 [==============================] - 0s 66ms/step - loss: 0.6853 - accuracy: 0.5490
2/2 [==============================] - 1s 17ms/step - loss: 0.6992 - accuracy: 0.5429
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_4 (Embedding)     (None, 45, 32)            3776      
                                                                 
 bidirectional (Bidirection  (None, 64)                16640     
 al)                                                             
                                                                 
 dense_8 (Dense)             (None, 32)                2080      
                                                                 
 dense_9 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 22529 (88.00 KB)
Trainable params: 22529 (88.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

In [16]:

```
model2.evaluate(x_test, y_test)
```

```
2/2 [==============================] - 0s 15ms/step - loss: 0.6992 - accuracy: 0.5429
```

Out[16]:

```
[0.699213981628418, 0.5428571701049805]
```

In [17]:

```
from keras.preprocessing.text import Tokenizer

t = Tokenizer()
# De fin i ng 4 doc men t L i sts
fit_text = ['Machine Learning Knowledge', 'Machine Learning',
'Deep Learning',
'Artificial Intelligence']
t.fit_on_texts(fit_text)
```