Notebook_Run

December 10, 2021

[1]:
```
# Importing Main Libraries For Empathy Prediction Problem Set
import torch
import numpy as np

torch.cuda.current_device()
from sklearn.model_selection import train_test_split
import torch.backends.cudnn as cudnn
import warnings
from torch.utils.data import DataLoader
from data_embeddings import EmpathyDataLoading, DataProcessing
from lstm_models import LSTM_fix_input, LSTM_var_input, LSTM_glove_vecs_input
from training_testing_criterion import train, get_criterion_optimizer_scheduler

warnings.filterwarnings('ignore')
np.random.seed(1)

# Model training using GPU in CUDA Environment
print("Whether Cuda is Available: {}".format(torch.cuda.is_available()))

# File Names
label_message_file = "/media/HDD_2TB.1/Empathy-Predictions/labeled_messages.csv"
empathies = "/media/HDD_2TB.1/Empathy-Predictions/empathies.csv"
```

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

Whether Cuda is Available: True
```

[2]:
```
# Method that calls to train different types of LSTMs
def training_LSTMs(model, epochs, learning_rate, loss_weights, device,␣
 ↪train_queue, valid_queue):
    model = model.to(device)
    criterion, optimizer, scheduler = get_criterion_optimizer_scheduler(model,␣
 ↪epochs, learning_rate,
                                                                        ␣
 ↪loss_weights, device)
```

```
    for epoch in range(epochs):
        scheduler.step()
        train(model, device, train_queue, valid_queue, optimizer, epoch,␣
↪criterion)
```

[3]:
```
# Main Method
def main():
    # Object 'Data' is created by Class Named -> 'DataProcessing'. file names␣
↪are parameters
    data_obj = DataProcessing(label_message_file, empathies)

    # Method describes messages lengths, number of words in Corpus
    data_obj.describe_counts()

    # Method for MultiLabel Encoding, weighted label weights for label data␣
↪imbalance
    output_size, loss_weights = data_obj.label_binarizer_get_weights()

    # Get X and Y
    X = data_obj.get_X_data()
    y = data_obj.get_Y_data()
    print('\n')

    # Training and Testing Split
    X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.3)

    # Baseline Classifier using Support Vector Classifier to calculate Accuracy␣
↪and Area Under Curve
    print("*** Baseline AUC Scores ***")
    acc_svm, roc_svm = data_obj.modelling("SVC", X_train, X_valid, y_train,␣
↪y_valid)
    print("SVM Modelling --> Validation Acc. : %.3f, Validation AUC Score : %.
↪3f" % (acc_svm, roc_svm))
    acc_RF, roc_RF = data_obj.modelling("RandomForest", X_train, X_valid,␣
↪y_train, y_valid)
    print("Random Forest Modelling --> Validation Acc. : %.3f, Validation AUC␣
↪Score : %.3f" % (acc_RF, roc_RF))
    print("** Statistical Method performed better then Baseline **")

    # 'EmpathyDataLoading' Class for training and validation data to load while␣
↪run time
    train_ds = EmpathyDataLoading(X_train, y_train)
    valid_ds = EmpathyDataLoading(X_valid, y_valid)

    vocab_size = len(data_obj.words)
    epochs = 1001
```

```
    batch_size = 1000
    learning_rate = 0.3

    # Data Loader for train and test
    train_queue = DataLoader(train_ds, batch_size=batch_size, shuffle=True)
    valid_queue = DataLoader(valid_ds, batch_size=batch_size, shuffle=False)

    # CUDA Environment Conf.
    torch.cuda.set_device(0)
    cudnn.benchmark = True
    torch.manual_seed(1)
    cudnn.enabled = True
    torch.cuda.manual_seed(1)
    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    # LSTMs models with fixed Input length, var Input length, using Stanford␣
↪Glove Vec Representations
    print('\n')
    print('-----------LSTMs Fixed Input Length--------------')
    model_fix_len = LSTM_fix_input(vocab_size, 48, 96, output_size)
    training_LSTMs(model_fix_len, epochs, learning_rate, loss_weights, device,␣
↪train_queue, valid_queue)

    print('\n')
    print('-----------LSTMs Var Input Length--------------')
    model_var_len = LSTM_var_input(vocab_size, 48, 96, output_size)
    training_LSTMs(model_var_len, epochs, learning_rate, loss_weights, device,␣
↪train_queue, valid_queue)

    print('\n')
    print('-----------LSTMs with Glove Representations--------------')
    word_vecs = data_obj.load_glove_vectors()
    pretrained_weights, vocab, vocab2index = data_obj.get_emb_matrix(word_vecs)
    model_glove = LSTM_glove_vecs_input(vocab_size, 50, 96, pretrained_weights,␣
↪output_size)
    training_LSTMs(model_glove, epochs, learning_rate, loss_weights, device,␣
↪train_queue, valid_queue)
```

[4]:
```
if __name__ == '__main__':
    main()
```

```
Data Shape is: (3562, 4)
Intial 10 Rows :
   num_seen          message       empathy ignore
0      2884            tired         tired    NaN
1       253        exhausted         tired    NaN
2        61          drained         tired    NaN
```

```
3        31  tired but happy  tired, happy    NaN
4        30         im tired         tired    NaN
5        29       very tired         tired    NaN
6        28      a bit tired         tired    NaN
7        28     i feel tired         tired    NaN
8        25   tired but good   tired, good    NaN
9        24         worn out         tired    NaN


Number of words in Corpus: 2114
Message Avg Length : 5.123526108927569, Message Max Length : 121
Number of Empathies: 62, Output Shape is: (3562, 62)


First 10 Columns of Data After Data Preprocessing and MultiLabel Encoding:
   num_seen          message       empathy  message_length  \
0      2884            tired         tired               1
1       253        exhausted         tired               1
2        61          drained         tired               1
3        31  tired but happy  tired, happy               3
4        30       i am tired         tired               3
5        29       very tired         tired               2
6        28      a bit tired         tired               3
7        28     i feel tired         tired               3
8        25   tired but good   tired, good               3
9        24         worn out         tired               2

                                              encoded       y_encoded  \
0  [[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...         [tired]
1  [[3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...         [tired]
2  [[4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...         [tired]
3  [[2, 5, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  [tired, happy]
4  [[7, 8, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...         [tired]
5  [[9, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...         [tired]
6  [[10, 11, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...         [tired]
7  [[7, 12, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...         [tired]
8  [[2, 5, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...   [tired, good]
9  [[14, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...         [tired]

                                        y_encoded_int
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
4  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
5  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
6  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
7  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

```
8  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
9  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

```
*** Baseline AUC Scores ***
SVM Modelling --> Validation Acc. : 0.976, Validation AUC Score : 0.584
Random Forest Modelling --> Validation Acc. : 0.900, Validation AUC Score :
0.698
** Statistical Method performed better then Baseline **
```

```
----------LSTMs Fixed Input Length--------------
Epoch: 1, Train loss: 0.676, Val loss: 0.668, Val Acc: 0.621, Val AUC_ROC Macro:
0.488, Val AUC_ROC Weighted : 0.488, Val Recall Macro: 0.476, Val Precision
Weighted : 0.955
Epoch: 101, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.636, Val AUC_ROC Weighted : 0.636, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 201, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.678, Val AUC_ROC Weighted : 0.678, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 301, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.697, Val AUC_ROC Weighted : 0.697, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 401, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.705, Val AUC_ROC Weighted : 0.705, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 501, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.708, Val AUC_ROC Weighted : 0.708, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 601, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.710, Val AUC_ROC Weighted : 0.710, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 701, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.712, Val AUC_ROC Weighted : 0.712, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 801, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.712, Val AUC_ROC Weighted : 0.712, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 901, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.713, Val AUC_ROC Weighted : 0.713, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 1001, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.713, Val AUC_ROC Weighted : 0.713, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
```

```
----------LSTMs Var Input Length--------------
Epoch: 1, Train loss: 0.673, Val loss: 0.666, Val Acc: 0.614, Val AUC_ROC Macro:
0.477, Val AUC_ROC Weighted : 0.477, Val Recall Macro: 0.486, Val Precision
Weighted : 0.956
Epoch: 101, Train loss: 0.011, Val loss: 0.011, Val Acc: 0.978, Val AUC_ROC
Macro: 0.479, Val AUC_ROC Weighted : 0.479, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 201, Train loss: 0.009, Val loss: 0.010, Val Acc: 0.978, Val AUC_ROC
Macro: 0.479, Val AUC_ROC Weighted : 0.479, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 301, Train loss: 0.009, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.481, Val AUC_ROC Weighted : 0.481, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 401, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.482, Val AUC_ROC Weighted : 0.482, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 501, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.484, Val AUC_ROC Weighted : 0.484, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 601, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.486, Val AUC_ROC Weighted : 0.486, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 701, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.487, Val AUC_ROC Weighted : 0.487, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 801, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.488, Val AUC_ROC Weighted : 0.488, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 901, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.488, Val AUC_ROC Weighted : 0.488, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 1001, Train loss: 0.008, Val loss: 0.009, Val Acc: 0.978, Val AUC_ROC
Macro: 0.488, Val AUC_ROC Weighted : 0.488, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957


----------LSTMs with Glove Representations--------------
Epoch: 1, Train loss: 0.684, Val loss: 0.677, Val Acc: 0.467, Val AUC_ROC Macro:
0.509, Val AUC_ROC Weighted : 0.509, Val Recall Macro: 0.473, Val Precision
Weighted : 0.955
Epoch: 101, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.615, Val AUC_ROC Weighted : 0.615, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 201, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.657, Val AUC_ROC Weighted : 0.657, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 301, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
```

```
Macro: 0.687, Val AUC_ROC Weighted : 0.687, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 401, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.700, Val AUC_ROC Weighted : 0.700, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 501, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.706, Val AUC_ROC Weighted : 0.706, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 601, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.707, Val AUC_ROC Weighted : 0.707, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 701, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.708, Val AUC_ROC Weighted : 0.708, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 801, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.708, Val AUC_ROC Weighted : 0.708, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 901, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.708, Val AUC_ROC Weighted : 0.708, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
Epoch: 1001, Train loss: 0.002, Val loss: 0.002, Val Acc: 0.978, Val AUC_ROC
Macro: 0.708, Val AUC_ROC Weighted : 0.708, Val Recall Macro: 0.500, Val
Precision Weighted : 0.957
```

[ ]: