CSC 219 - Fall 2023

Machine Learning

Project 4

Multi-Modal Deep Learning for Fake News Detection

Due at 3:00 pm, Monday,November 13, 2023

Team Members

Charitha Vadamala- 303284072

Navya Krishna Batchu-303188769

Problem statement:

The goal of this research is to develop a multi-modal deep learning system for automatically detecting false information on social media. Our technique aims to offer a potent instrument for detecting manipulation and false information in web multimedia content by merging textual analysis of tweets with visual content analysis of related photos. By helping media professionals and users with the crucial duty of verifying social media information, this novel approach may improve the accuracy of detecting false news and contribute to a more dependable and trustworthy online information environment.

Dataset description:

The dataset for this project is a huge multimodal fake news dataset with over 1 million data pieces. These samples, which come from a variety of sources, include a variety of data types like text, photos, metadata, and comments. The dataset is organized to support two-, three-, and six-way classification tasks, among others. The project only considers multimodal samples, that is, data that include both textual and picture information for the particular job at hand. The TSV dataset files' "clean_title" column is used to extract text data, while the "image_url" column is used to link images. Using subsets of the dataset is advised to speed up the training process, particularly for big volumes. Additionally, picture URLs which are provided in, can be used to handle the significant image data.

Methodology:

```
print("Training Data:", train_images.shape, train_texts.shape, train_labels.shape)
print("Test Data:", test_images.shape, test_texts.shape, test_labels.shape)
```

```
Training Data: (823, 224, 224, 3) (823, 100) (823, 1)
Test Data: (669, 224, 224, 3) (669, 100) (669, 1)
```

Multi Modal with Self Attention

```
21/21 [==============================] - 1s 21ms/step
              precision    recall  f1-score   support

           0       0.11      0.00      0.01       336
           1       0.49      0.98      0.65       333

    accuracy                           0.49       669
   macro avg       0.30      0.49      0.33       669
weighted avg       0.30      0.49      0.33       669
```

Multi Modal with Self Attention + VGG16

```
Epoch 12/20
26/26 [==============================] - 13s 515ms/step - loss: 0.0662 - accuracy: 1.0000 - val_loss: 0.6356 - val_accuracy: 0.6308
Epoch 13/20
26/26 [==============================] - 13s 516ms/step - loss: 0.0513 - accuracy: 1.0000 - val_loss: 0.6322 - val_accuracy: 0.6271
Epoch 14/20
26/26 [==============================] - 13s 514ms/step - loss: 0.0453 - accuracy: 1.0000 - val_loss: 0.6414 - val_accuracy: 0.6345
Epoch 15/20
26/26 [==============================] - 13s 517ms/step - loss: 0.0384 - accuracy: 1.0000 - val_loss: 0.6548 - val_accuracy: 0.6308
Epoch 16/20
26/26 [==============================] - 13s 515ms/step - loss: 0.0385 - accuracy: 1.0000 - val_loss: 0.6585 - val_accuracy: 0.6330
Epoch 17/20
26/26 [==============================] - 13s 517ms/step - loss: 0.0322 - accuracy: 1.0000 - val_loss: 0.6822 - val_accuracy: 0.6308
Epoch 18/20
26/26 [==============================] - 13s 516ms/step - loss: 0.0253 - accuracy: 1.0000 - val_loss: 0.6962 - val_accuracy: 0.6368
Epoch 19/20
26/26 [==============================] - 13s 516ms/step - loss: 0.0265 - accuracy: 1.0000 - val_loss: 0.7224 - val_accuracy: 0.6442
Epoch 20/20
26/26 [==============================] - 13s 516ms/step - loss: 0.0229 - accuracy: 1.0000 - val_loss: 0.7505 - val_accuracy: 0.6271
21/21 [==============================] - 3s 126ms/step - loss: 0.7505 - accuracy: 0.6271
```

[12]:
```
[0.7505105137825012, 0.627055287361145]
```

[13]:
```
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report

# Adjust the threshold for binary classification
test_pred = model_msa1.predict([test_images, test_texts])
test_pred = np.argmax(test_pred.reshape(-1, 2), axis=1)

# Now use them in classification_report
print(classification_report(test_labels, test_pred))
```

```
21/21 [==============================] - 3s 125ms/step
              precision    recall  f1-score   support

           0       0.51      0.65      0.57       336
           1       0.51      0.36      0.42       333

    accuracy                           0.51       669
   macro avg       0.51      0.51      0.50       669
weighted avg       0.51      0.51      0.50       669
```

Multi Modal with Co-Attention

```
Epoch 2/10
26/26 [==============================] - 2s 66ms/step - loss: 0.7221 - accuracy: 0.5869 - val_loss: 0.6867 - val_accuracy: 0.5501
Epoch 3/10
26/26 [==============================] - 2s 66ms/step - loss: 0.6856 - accuracy: 0.6330 - val_loss: 0.6837 - val_accuracy: 0.6054
Epoch 4/10
26/26 [==============================] - 2s 67ms/step - loss: 0.6163 - accuracy: 0.6598 - val_loss: 0.6809 - val_accuracy: 0.6248
Epoch 5/10
26/26 [==============================] - 2s 66ms/step - loss: 0.6280 - accuracy: 0.6646 - val_loss: 0.6780 - val_accuracy: 0.6024
Epoch 6/10
26/26 [==============================] - 2s 66ms/step - loss: 0.5747 - accuracy: 0.7035 - val_loss: 0.6751 - val_accuracy: 0.6383
Epoch 7/10
26/26 [==============================] - 2s 75ms/step - loss: 0.5741 - accuracy: 0.7047 - val_loss: 0.6711 - val_accuracy: 0.6054
Epoch 8/10
26/26 [==============================] - 2s 67ms/step - loss: 0.5412 - accuracy: 0.7254 - val_loss: 0.6687 - val_accuracy: 0.5934
Epoch 9/10
26/26 [==============================] - 2s 66ms/step - loss: 0.5234 - accuracy: 0.7497 - val_loss: 0.6628 - val_accuracy: 0.6413
Epoch 10/10
26/26 [==============================] - 2s 66ms/step - loss: 0.4975 - accuracy: 0.7485 - val_loss: 0.6581 - val_accuracy: 0.6712
21/21 [==============================] - 0s 19ms/step - loss: 0.6581 - accuracy: 0.6712
```

[63]:
```
[0.6580588221549988, 0.6711509823799133]
```

[64]:
```
from sklearn.metrics import precision_score, recall_score, f1_score,classification_report
# Adjust the threshold for binary classification
test_pred = model.predict([test_images, test_texts])


threshold = 0.5
test_pred = (test_pred > threshold).astype(int)

print(classification_report(test_labels, test_pred))
```

```
21/21 [==============================] - 1s 17ms/step
              precision    recall  f1-score   support

           0       0.72      0.57      0.63       336
           1       0.64      0.78      0.70       333

    accuracy                           0.67       669
   macro avg       0.68      0.67      0.67       669
weighted avg       0.68      0.67      0.67       669
```

Multi Modal with Co-Attention + ResNet

```
26/26 [==============================] - 10s 374ms/step - loss: 0.8043 - accuracy: 0.5383 - val_loss: 0.7061 - val_accuracy: 0.4978
Epoch 3/10
26/26 [==============================] - 10s 377ms/step - loss: 0.7905 - accuracy: 0.5614 - val_loss: 0.7079 - val_accuracy: 0.4978
Epoch 4/10
26/26 [==============================] - 10s 378ms/step - loss: 0.7616 - accuracy: 0.5699 - val_loss: 0.7092 - val_accuracy: 0.4978
Epoch 5/10
26/26 [==============================] - 10s 380ms/step - loss: 0.7540 - accuracy: 0.5905 - val_loss: 0.7012 - val_accuracy: 0.4978
Epoch 6/10
26/26 [==============================] - 10s 382ms/step - loss: 0.6964 - accuracy: 0.6051 - val_loss: 0.6939 - val_accuracy: 0.5022
Epoch 7/10
26/26 [==============================] - 10s 382ms/step - loss: 0.6776 - accuracy: 0.6100 - val_loss: 0.6942 - val_accuracy: 0.5082
Epoch 8/10
26/26 [==============================] - 10s 384ms/step - loss: 0.6930 - accuracy: 0.6136 - val_loss: 0.7051 - val_accuracy: 0.5007
Epoch 9/10
26/26 [==============================] - 10s 386ms/step - loss: 0.6560 - accuracy: 0.6367 - val_loss: 0.7884 - val_accuracy: 0.5022
Epoch 10/10
26/26 [==============================] - 10s 388ms/step - loss: 0.6361 - accuracy: 0.6355 - val_loss: 0.6999 - val_accuracy: 0.5232
21/21 [==============================] - 2s 96ms/step - loss: 0.6999 - accuracy: 0.5232
```

[39]:
```
[0.6998950242996216, 0.5231689214706421]
```

[40]:
```python
from sklearn.metrics import precision_score, recall_score, f1_score,classification_report
# Adjust the threshold for binary classification
test_pred = model.predict([test_images, test_texts])


threshold = 0.5
test_pred = (test_pred > threshold).astype(int)

print(classification_report(test_labels, test_pred))
```

```
21/21 [==============================] - 3s 94ms/step
              precision    recall  f1-score   support

           0       0.52      0.82      0.63       336
           1       0.55      0.22      0.31       333

    accuracy                           0.52       669
   macro avg       0.53      0.52      0.47       669
weighted avg       0.53      0.52      0.48       669
```

Multi Modal Cross Attention

```
Epoch 4/10
26/26 [==============================] - 2s 76ms/step - loss: 0.5471 - accuracy: 0.7139 - val_loss: 0.6865 - val_accuracy: 0.5605
Epoch 5/10
26/26 [==============================] - 2s 74ms/step - loss: 0.5142 - accuracy: 0.7448 - val_loss: 0.6840 - val_accuracy: 0.5605
Epoch 6/10
26/26 [==============================] - 2s 74ms/step - loss: 0.4668 - accuracy: 0.7789 - val_loss: 0.6834 - val_accuracy: 0.5486
Epoch 7/10
26/26 [==============================] - 2s 75ms/step - loss: 0.4142 - accuracy: 0.8153 - val_loss: 0.6760 - val_accuracy: 0.5912
Epoch 8/10
26/26 [==============================] - 2s 76ms/step - loss: 0.3921 - accuracy: 0.8226 - val_loss: 0.6735 - val_accuracy: 0.5987
Epoch 9/10
26/26 [==============================] - 2s 75ms/step - loss: 0.3604 - accuracy: 0.8621 - val_loss: 0.6728 - val_accuracy: 0.5979
Epoch 10/10
26/26 [==============================] - 2s 75ms/step - loss: 0.3406 - accuracy: 0.8597 - val_loss: 0.6780 - val_accuracy: 0.5717
21/21 [==============================] - 0s 21ms/step - loss: 0.6780 - accuracy: 0.5717
```

[60]:
```
[0.6780408024787903, 0.5717488527297974]
```

[62]:
```python
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report

# Adjust the threshold for binary classification
test_pred = model_cross.predict([test_images, test_texts])

test_pred_last_step = test_pred[:, -1, :]

test_pred_flat = test_pred_last_step.flatten()

threshold = 0.5
test_pred_binary = (test_pred_flat > threshold).astype(int)

print(classification_report(test_labels, test_pred_binary))
```

```
21/21 [==============================] - 1s 20ms/step
              precision    recall  f1-score   support

           0       0.77      0.18      0.30       336
           1       0.53      0.94      0.68       333

    accuracy                           0.56       669
   macro avg       0.65      0.56      0.49       669
weighted avg       0.65      0.56      0.49       669
```

Multi Modal Cross Attention + ResNet50

```
26/26 [==============================] - 10s 401ms/step - loss: 0.1257 - accuracy: 0.9836 - val_loss: 0.7261 - val_accuracy: 0.5105
Epoch 7/10
26/26 [==============================] - 10s 397ms/step - loss: 0.1096 - accuracy: 0.9878 - val_loss: 0.7319 - val_accuracy: 0.5142
Epoch 8/10
26/26 [==============================] - 10s 397ms/step - loss: 0.0903 - accuracy: 0.9921 - val_loss: 0.7442 - val_accuracy: 0.5157
Epoch 9/10
26/26 [==============================] - 10s 395ms/step - loss: 0.0766 - accuracy: 0.9957 - val_loss: 0.7510 - val_accuracy: 0.5194
Epoch 10/10
26/26 [==============================] - 10s 394ms/step - loss: 0.0626 - accuracy: 1.0000 - val_loss: 0.7453 - val_accuracy: 0.5314
21/21 [==============================] - 2s 97ms/step - loss: 0.7453 - accuracy: 0.5314
```

[53]:
```
[0.745265007019043, 0.5313901305198669]
```

[54]:
```
print(test_labels.shape, test_pred.shape)
```

```
(669, 1) (669, 1)
```

[59]:
```
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report

# Adjust the threshold for binary classification
test_pred = model.predict([test_images, test_texts])

test_pred_last_step = test_pred[:, -1, :]

test_pred_flat = test_pred_last_step.flatten()

threshold = 0.5
test_pred_binary = (test_pred_flat > threshold).astype(int)

print(classification_report(test_labels, test_pred_binary))
```

```
21/21 [==============================] - 2s 97ms/step
              precision    recall  f1-score   support

           0       0.62      0.16      0.25       336
           1       0.52      0.90      0.66       333

    accuracy                           0.53       669
   macro avg       0.57      0.53      0.45       669
weighted avg       0.57      0.53      0.45       669
```

Task Division and Project Reflection:

Self Attention and VGG 16 : Charitha Vadamala

Co Attention and ResNet : Navya Krishna Batchu

Cross Attention and ResNet 50 : Charitha Vadamala

Proposal and Problem Statement : Navya Krishna Batchu

Screenshots and Tabulation : Charitha Vadamala

Additional Features :

BERT - Self Attention

```
==================================================================================================
Total params: 149161665 (569.01 MB)
Trainable params: 149161665 (569.01 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

[27]:
```python
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report


# Adjust the threshold for binary classification
test_pred = model.predict([test_images, test_texts])

threshold = 0.5
test_pred = (test_pred > threshold).astype(int)

# Now use them in classification_report
print(classification_report(test_labels, test_pred))

precision = precision_score(test_labels, test_pred)
recall = recall_score(test_labels, test_pred)
f1 = f1_score(test_labels, test_pred)

print("Precision: {:.2f}".format(precision))
print("Recall: {:.2f}".format(recall))
print("F1 Score: {:.2f}".format(f1))
```

```
30/30 [==============================] - 5s 168ms/step
              precision    recall  f1-score   support

           0       0.60      0.50      0.54       468
           1       0.57      0.67      0.62       465

    accuracy                           0.58       933
   macro avg       0.59      0.58      0.58       933
weighted avg       0.59      0.58      0.58       933

Precision: 0.57
Recall: 0.67
F1 Score: 0.62
```

- ●