Check-In 3

Introduction

Our project is based off of a paper implementing a relational context learning and multiplex fusion network to address the task of multimodal sarcasm detection - classifying several modes of inputs as sarcastic or not. This classification task is important because it can help filter misinformation online, and is also helpful for other machine learning tasks such as sentiment analysis. The paper focuses on implementing sarcasm detection *without* the use of graph structures, which is used by many existing sarcasm detection models. Using graph structures presents several limitations - most importantly, it is extremely computationally expensive to create graph networks modeling images and text. Instead, the model aims to understand dynamic relationships between the image and text pairs to capture the context needed to classify something as sarcastic or not. We chose this paper because it addresses a problem that we are all interested in and it builds off of concepts we've discussed previously in class - such as various types of attention.

Challenges

One of the hardest challenges of the project thus far was initially figuring out how to process our training data. We needed the data in two formats for the CLIP/Resnet and BERT models. The paper implemented a complex custom dataset that was hard to follow, but we were ultimately able to process the data without implementing this custom class.

Another problem we ran into was not being able to train the CLIP parameters of our model. When running on Oscar, we kept on running into out-of-memory errors. We decided to just freeze all of the parameters of CLIP as well.

Finally, one bug we ran into with our model was that it trained to only classify image-text pairs as sarcastic. After some debugging, we realized that this was due to the training data being shuffled within each batch only - so this only shuffled every increment of 32 data points. After adjusting our data shuffling to shuffle throughout the entire dataset, we were able to achieve higher accuracy with our model and predictions of both 0's and 1's.

Insights

We've trained some initial versions of our model (some with bugs) and have recorded the train/test results below:

1. With buggy model, CLIP, bert frozen, 5 epochs:
    a. Training
        i.

Epoch	Accuracy	Precision	Recall	F1 Score
0	0.90	0.94	0.86	0.90
1	0.92	0.97	0.86	0.91
2	0.92	0.96	0.88	0.92
3	0.92	0.95	0.87	0.91
4	0.92	0.96	0.87	0.91

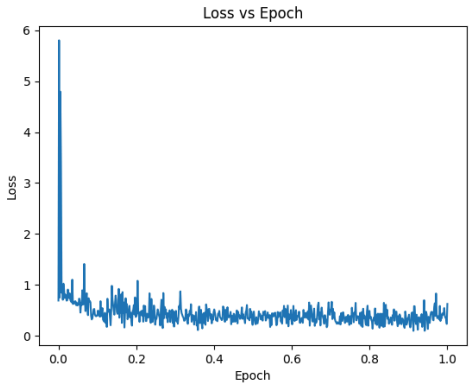
    b. Testing (rip)
        i. Accuracy: 0.4305

ii. Precision: 0.4305
iii. Recall: 1.0000
iv. F1 Score: 0.6019

2. With fixed model, BERT, Resnet frozen, clip frozen (still shuffling data badly)
    a. Training (2 epochs)
        i.

Epoch	Accuracy	Precision	Recall	F1 Score
0	0.93			0.92
1	0.95			0.94

    b. Testing
        i. Accuracy: 0.44
        ii. Precision: 0.44
        iii. Recall: 1.0 (classifying all as 1's)
        iv. F1 Score: 0.61
3. Shuffling training data overall, not just within batches; BERT/Resnet/CLIP frozen
    a. Training (1 epoch)
        i. Accuracy: 0.81
        ii. F1: 0.80
        iii. Loss:
        1.



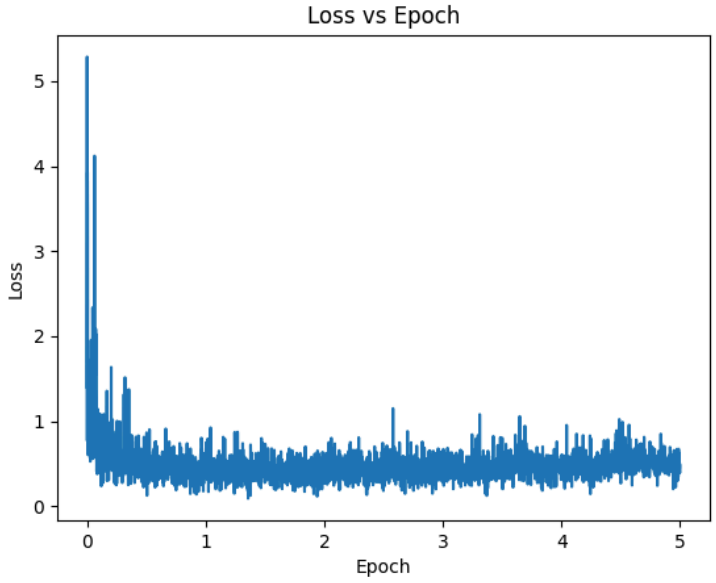
    b. Testing
        i. MMSD 2.0
            1. Accuracy 0.72
            2. Precision 0.61
            3. Recall 0.95
            4. F1 Score 0.74
        ii. MUSE (potentially a bad dataset to test on given our previous bug bc it contains only sarcastic data)
            1. Accuracy: 0.98
            2. Precision: 1.0
            3. Recall: 0.98

4. F1 Score: 0.99

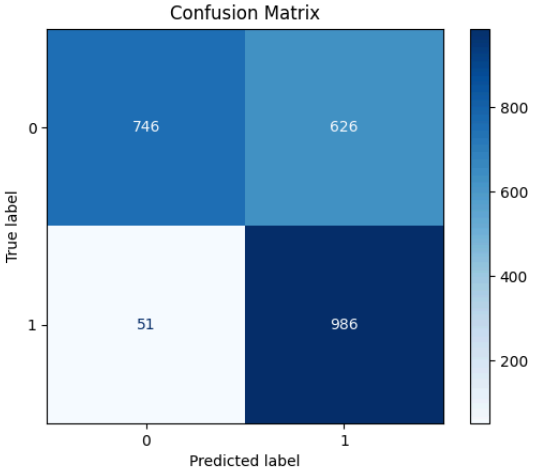
4. 5 epochs of the updated model
    a. Train
        i.

Epoch	Accuracy	Precision	Recall	F1 Score
1	0.77	0.76	0.76	0.76
2	**0.82**	**0.81**	**0.82**	**0.82**
3	0.81	0.80	0.81	0.80
4	0.80	0.79	0.80	0.80
5	0.76	0.75	0.76	0.75

        ii. Loss



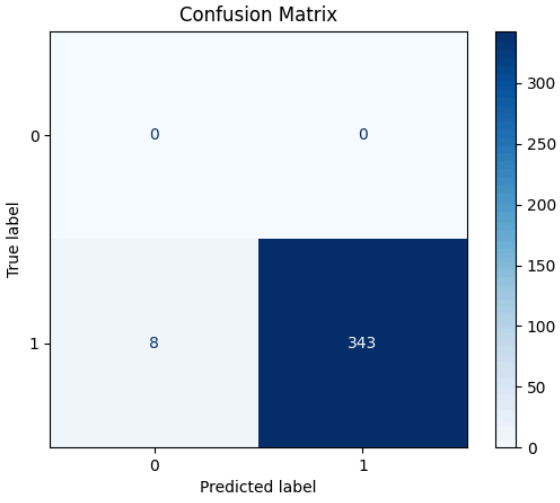
        iii.
    b. Test (using weights after 5 epochs)
        i. MMSD 2.0
            1. Accuracy: 0.72
            2. Precision: 0.61
            3. Recall: 0.95
            4. F1 Score: 0.74
            5. Confusion matrix



a.

ii. MUSE

1. Accuracy: 0.98
2. Precision: 1.0
3. Recall: 0.98
4. F1 Score: 0.99
5. Confusion matrix



a.

Plan

We are on track! We have tried training several different iterations of our model, always achieving a high training accuracy and low testing accuracy, but working towards a higher test accuracy. We are working through debugging these issues and trying to figure out how to make our model more accurate, as well as training our model for more epochs. More time needs to be dedicated to preprocessing our new dataset and testing our model on non-MMSD data. We are thinking of changing some of our Tensorflow builtin layers to more accurately reflect the RCLMuFN model and hopefully increase our accuracy on testing data. Additionally, we may try a few more strategies that would allow us to unfreeze

the CLIP weights during training (higher learning rate, more GPUs, etc) and potentially make our model more applicable to test and validation data.