CSE556
Natural Language Processing
Assignment III

Aaradhya Verma 2022004
Aditya Bagri 2022028
Mudit Bansal 2022306

Task I

Pre-processing:

For preprocessing, we essentially flatten the input, convert it to lower - case and store all the words, we construct a word level tokenizer storing all the words mapped to their respective embeddings and an inverse map for embedding to word mappings. The tokenizer also consists of special tokens like [START], [END], [PAD], [UNK]

Model Architecture:

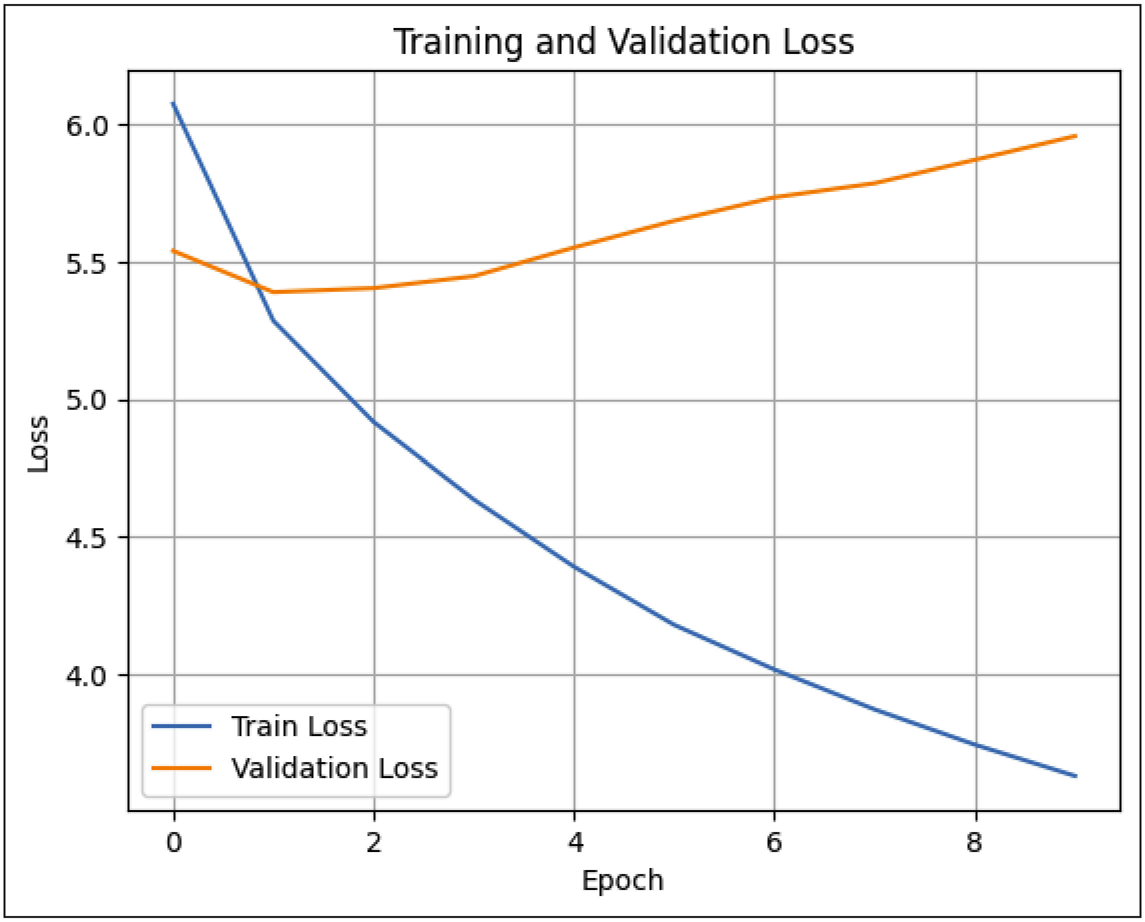
```
TransformerLM(
  (token_embedding): Embedding(11641, 128)
  (positional_encoding): PositionalEncoding(
    (pos_embedding): Embedding(512, 128)
  )
  (dropout): Dropout(p=0.2, inplace=False)
  (decoder_layers): ModuleList(
    (0): TransformerBlock(
      (self_attn): MultiHeadAttention(
        (W_q): Linear(in_features=128, out_features=128, bias=True)
        (W_k): Linear(in_features=128, out_features=128, bias=True)
        (W_v): Linear(in_features=128, out_features=128, bias=True)
        (W_o): Linear(in_features=128, out_features=128, bias=True)
      )
      (feed_forward): PositionWiseFeedForward(
        (linear1): Linear(in_features=128, out_features=256, bias=True)
        (linear2): Linear(in_features=256, out_features=128, bias=True)
      )
      (norm1): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.2, inplace=False)
    )
  )
  (fc_out): Linear(in_features=128, out_features=11641, bias=True)
)
```

We use the nn.Embedding layer to create token embeddings, then we use a positional encoding layer to take the position of the tokens into account. Then we use a dropout layer to perform regularisation. We use 1 Decoder Style Transformer Block consisting of a Multi Head Attention block with 2 heads, a Position Wise FFN, Layer Norms and a dropout layer. And in the end we use a fully connected layer to generate token probabilities.

Hyper-Parameters:

1. Vocab Size - 13393
2. Embedding Dimension - 128
3. Number of Heads - 2
4. Number of Layers - 1
5. Dimension of FFN - 256
6. Dropout - 0.2
7. Learning Rate - 2e-3
8. Weight Decay - 1e-4
9. Label Smoothing - 0.01
10. Number of Epochs - 10

Training and Validation Loss Plots:



As we can see, the model starts overfitting after the 2nd Epoch.

Perplexity Score on validation sets:

```
100%|██████████| 308/308 [00:31<00:00,  9.89it/s]

Epoch 2: Train Loss = 5.2844, Val Loss = 5.3884, Perplexity Score Train = 197.2354, Preplexity Score Validation = 218.8594
Sample text: thou art is the most beautiful of my poor good heart , that that she will not a man in his name ?
 : i hear for heaven 's right ; and his majesty .
 with
```

Task II

Pre-Processing

An acronym and contraction dictionary was made by going through the dataset beforehand
Steps:

- All the characters are converted to lowercase letters
- Contractions library is used to expand any common contractions and slangs used in language
- Regex is used to find words which belong to the expansion dictionary and are replaced by their expansions
- Regex is used to remove all strings which start with www. Http https , removing almost all links
- Regex is used to remove any special character except for numbers and alphabets
- Regex is used to remove extra whitespace

Pre processed dataset is then stored as a csv file which can be loaded in further training

Sample function from data frames is used to split the data into 70% train , 15% val , 15% test

Model Architecture

T5

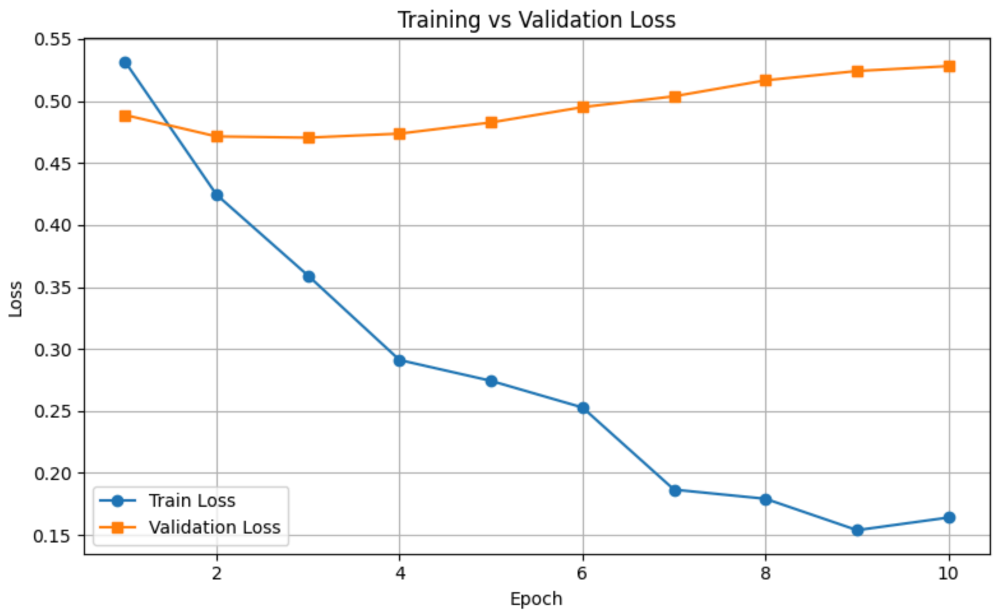
T5 base is used for tokenization and for generation
Before fine tuning the dataset was preprocessed by adding the word normalize before the input text , essentially giving it input as Normalize <input text> , and the label as the given claim all padded to the same lengths

Training:

- For training the trainer class was used
- The learning rate was set to 3e-4

- Model was trained for 10 epochs
- Weight decay of 0.1 was added to prevent overfitting
- Batch size of 8 was used
- The best model was loaded for evaluation at the end of training
- Data collater was used to pad input and labels to the same length

Train and Val Loss plots for T5



Evaluation Metrics for T5

```
ROUGE-L Precision: 0.4087
ROUGE-L Recall:    0.4325
ROUGE-L F1 Score:  0.3975
BLEU Score:       0.2125
BERT Score F1:    0.8901
```

Bart

Bart large is used for tokenization and for generation

Before fine tuning the dataset was preprocessed by tokenizing and padding

Training:

- For training the trainer class was used
- The learning rate was set to 3e-5
- Model was trained for 10 epochs

- Weight decay of 0.1 was added to prevent overfitting
- Batch size of 8 was used
- The best model was loaded for evaluation at the end of training
- Data collater was used to pad input and labels to the same length

<u>Train and Val Loss plots for Bart</u>

<u>Evaluation Metrics for Bart</u>

```
ROUGE-L Precision: 0.4855
ROUGE-L Recall:    0.4419
ROUGE-L F1 Score:  0.4400
BLEU Score:       0.2727
BERT Score F1:    0.9011
```

<u>Comparative analysis</u>

<u>ROUGE-L Precision & F1:</u> BART-large shows noticeably higher precision (0.4855 vs. 0.4087) and F1 (0.4400 vs. 0.3975), indicating that its summaries capture a higher proportion of the important content relative to the generated tokens.

BLEU Score: A BLEU score improvement (0.2727 vs. 0.2125) for BART-large suggests that its generated text matches reference texts more closely at the n-gram level.

BERT Score F1: With a score of 0.9011 compared to T5-base's 0.8901, BART-large achieves better semantic similarity between its outputs and the reference normalized claims.

Overall Bart outperforms T5

Resource Constraints and Model Selection:

When selecting models for fine-tuning on the claim normalization task, a key consideration was the balance between performance and available computational resources.

- T5-base offers a lower parameter count and reduced memory requirements, making it an attractive option when GPU memory and training time are limited.
- As the Dataset was limited going for a bigger model would have probably overfit and given the GPU requirements would have taken a lot longer to train

- Bart large was a bigger model but the training time was not significantly more , with the evaluation metrics being better for Bart large it was chosen

Task III

- Importing necessary libraries
  Python libraries:
    - os, numpy, matplotlib, pandas,
    - torch, pickle, torchvision, csv, PIL,
    - transformers, tqdm, evaluate & nltk

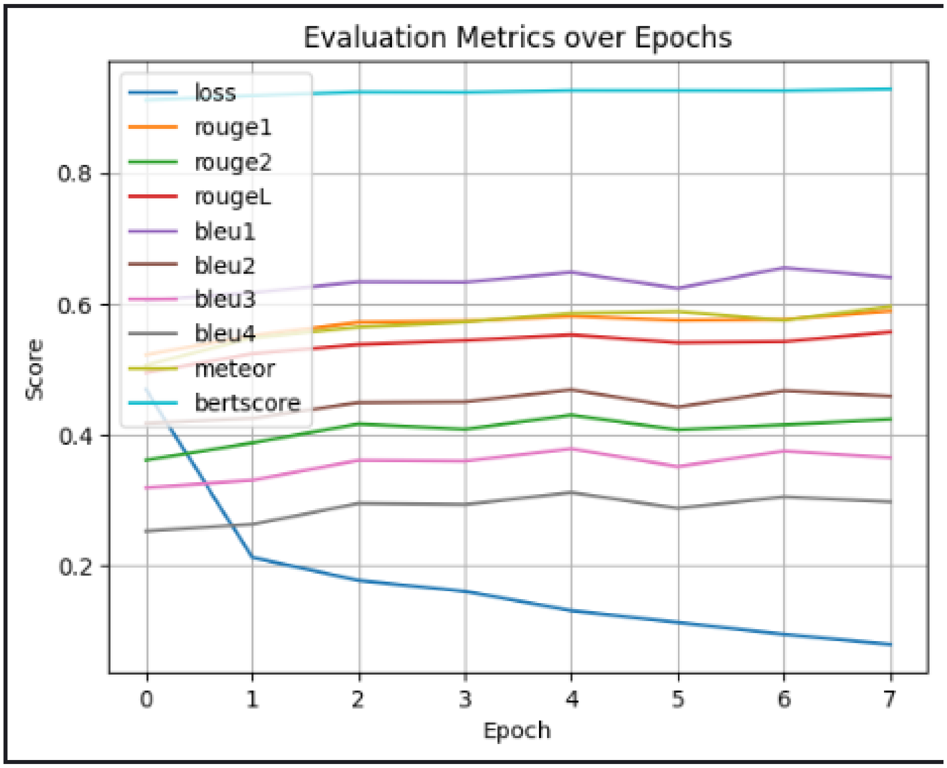
  are imported
- Loading train and validation:
    - Data Frames,
    - Image Descriptions
    - Objects Detected

  are loaded
- Instances of
    - Tokenizer
    - BART Model
    - VIT Model
    - VIT Processor

  are created from pre-trained methods
- Define the custom dataset & dataloader instances of train and validation
- Intialise *rouge*, *bleu*, *metoer* & *bertscore* instances for metric calculation. And initialise the metric-history dictionary
- Define TURBOModule, following the architecture mentioned in the paper. Notably, Knowledge Gates and GCNs are omitted as directed.
- Create a TURBOModule instance using BART and ViT Model instances.
- Create an instance of AdamW optimizer with *learning_rate* of **3e-5**
- Define method for plotting the metrics
- Define the training loop:
  - set checkpoint directory
  - training is looped over *num_epochs* times
  - at every epoch, each batch is iterated over, and model is used to extract predictions
  - Then, loss is calculated and and a *backward* step is initialised
  - Model is saved for the epoch
  - The trained model is then evaluated over the *validation* dataset and metrics are calculated, with *metrics_history* is updated.

  Additionally, based on the ‘RougeL’ score, the best performing model is updated. Finally, in a case where teh model doesn’t improve after *early_stopping_patience*, the model stops learning.

- Plot the metrics




```
Training loss: 0.0797

Validation Metrics:
 - ROUGE-1: 0.5888
 - ROUGE-2: 0.4235
 - ROUGE-L: 0.5564
 - BLEU-1 to BLEU-4: [0.6403, 0.4584, 0.3645, 0.2975]
 - METEOR: 0.5961
 - BERTScore (F1): 0.9282
```

- Hyperparameters:
  - learning rate = 3e-5
  - maximum length = 128
  - hidden dimension = 768
- Sample Validation predicting:

```
Example Predictions:

Original Post     : '<user> thank u for this awesome network in malad ( see pic ) .  # patheticcs'
[SEP] Sarcasm Target: <user>'s network in malad [SEP] Image Description:  [SEP] Detected Objects:
Ground Truth Exp : the author is pissed at <user> for not getting network in malad.
Predicted Exp     : the author is pissed at <user> for such awful network in malad.
------------------------------------------------------------
```

```
Original Post     : Nothing like waiting for an hour on the tarmac for a gate to come open in
snowy, windy Chicago! [SEP] Sarcasm Target: gate not opening  [SEP] Image Description:  [SEP]
Detected Objects:
Ground Truth Exp : nothing worst than waiting for an hour on the tarmac for a gate to come open in
snowy, windy chicago.
Predicted Exp    : nothing like waiting for an hour on the tarmac for a gate to come not opening
in snowy, windy Chicago.
------------------------------------------------------------
Original Post     : 'ahhh ! my favorite thing about spring ! # dst # springforward ' [SEP] Sarcasm
Target: spring weather [SEP] Image Description:  [SEP] Detected Objects:
Ground Truth Exp : nobody likes getting one hour of their life sucked away.
Predicted Exp    : the author hates spring.
------------------------------------------------------------
Original Post     : 'can anyone of you imagine a better way to start the new week than having a
salivary gland biopsy on monday morning ? me neither ...  emoji_300' [SEP] Sarcasm Target: having
a salivary gland biopsy [SEP] Image Description:  [SEP] Detected Objects:
Ground Truth Exp : having a salivary gland biopsy on monday morning is not a good way to start the
new week.
Predicted Exp    : having a salivary gland biopsy on monday morning isn't a good way to start the
new week.
------------------------------------------------------------
Original Post     : phew ! it 's going to be scorching hot this w-end !  the high on saturday is -
1 ! windchill will prob be - 30 . emoji_619 emoji_300 [SEP] Sarcasm Target: freezing weather [SEP]
Image Description:  [SEP] Detected Objects:
Ground Truth Exp : the author is worried that the weekend is going to be freezing with a high of
-1 and windchill probably -30.
Predicted Exp    : it's going to be freezing weather this w-end, the high on saturday is - 1,
windchill will prob be - 30.
------------------------------------------------------------
```

References

Task I

- https://medium.com/data-science/build-your-own-transformer-from-scratch-using-pytorch-84c850470dcb
- https://www.kaggle.com/code/arunmohan003/transformer-from-scratch-using-pytorch
- https://medium.com/@sayedebad.777/building-a-transformer-from-scratch-a-step-by-step-guide-a3df0aeb7c9a
- https://www.mislavjuric.com/transformer-from-scratch-in-pytorch/

Task II

Task III

- Mutli Head Attention
- Goel, P., Chauhan, D. S., & Akhtar, M. S. (2025, February 11).

  Target-Augmented Shared Fusion-based Multimodal Sarcasm Explanation Generation. arXiv.org. https://arxiv.org/abs/2502.07391

Thank you!