NLP Assignment 3

Manan Aggarwal 2022273
Shobhit Raj 2022482
Souparno Ghose 2022506

April 6, 2025

1 Task - Implement Transformer from Scratch

The saved model checkpoints for this task can be accessed here

1.1 Data Preprocessing and Tokenization

To prepare Shakespearean text for training, the following preprocessing steps were applied:

1. **Tokenization and Vocabulary:** Sentences are split into word tokens. A vocabulary of the 10,000 most frequent tokens is built.
2. **Special Tokens:** Four reserved tokens are added: `<PAD>` (0), `<START>` (1), `<STOP>` (2), and `<UNK>` (3).
3. **Tokenizer:** A word-to-index dictionary (`tokenizer`) and its inverse (`tokenizer_inv`) are created.
4. **Encoding:** Sentences are tokenized, a `<STOP>` token is added, and sequences are padded/truncated to 256 tokens.
5. **Dataset Wrapping:** The processed data is stored using a PyTorch `Dataset` class for easy batching.
6. **Utilities:** Functions for tokenizing, decoding, and padding help maintain consistent input formats.

This ensures standardized input for effective Transformer training and evaluation.

1.2 Model Architecture and Hyperparameters

The Transformer model integrates key components for effective autoregressive text generation:

1. **Embedding and Positional Encoding:**
   Tokens are mapped to continuous vectors via an embedding layer with dropout. Sinusoidal positional encodings (via the `PositionalEncoding` class) inject sequence order, and weight tying between the embedding and output projection (`fc_out`) reduces parameters and improves generalization.
2. **Self-Attention and Multi-Head Attention:**
   The `MultiHeadAttention` module:
   - Computes linear projections for queries (Q), keys (K), and values (V).
   - Splits projections into multiple heads (`split_heads_qkv()`).
   - Applies scaled dot-product attention with softmax normalization.

- Merges heads and applies a final linear projection (`W_o`).

3. **Causal Masking:**
   A causal mask (via `make_causal_mask()` and `apply_causal_mask()`) prevents attending to future tokens, ensuring autoregressive behavior.

4. **Transformer Blocks:**
   Each block uses pre-layer normalization, self-attention with residual connections and dropout, followed by a two-layer feed-forward network with GELU activation and a second residual connection.

5. **Regularization and Initialization:**
   Dropout is applied at multiple stages, and Xavier Normal initialization is used for stable training.

Table 1: Hyperparameters Used

Hyperparameter	**Value**
Batch Size	16
Optimizer	Adam
Learning Rate	1×10^{-3}
Epochs	50
Embedding Dimension	512
Number of Transformer Layers	6
Number of Heads	8
Max Length	256
Vocab Size	10000

This concise architecture effectively combines embeddings, positional encoding, multi-head self-attention, and feed-forward networks with residual connections and normalization to generate Shakespearean text.

1.3 Evaluation Metrics and Losses

We used cross-entropy loss (ignoring padding tokens) as our training objective and measured performance using both loss and perplexity.

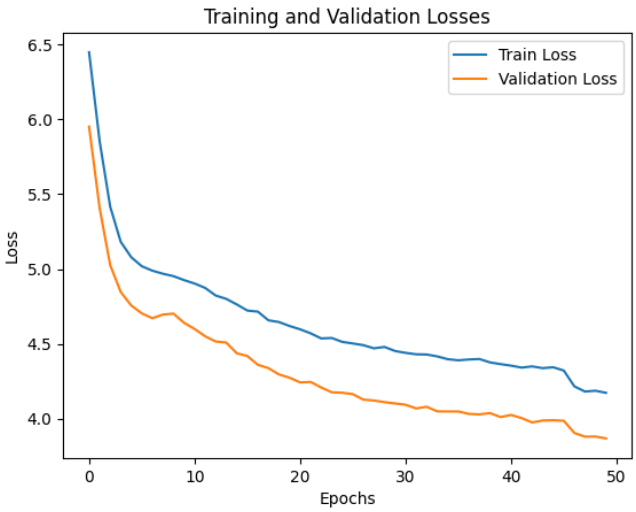

Figure 1: Training and Validation Losses

Table 2: Losses at Every 5 Epochs

Epoch	**Train Loss**	**Validation Loss**
5	5.0788	4.7563
10	4.9263	4.6408
15	4.7636	4.4369
20	4.6193	4.2738
25	4.5132	4.1732
30	4.4522	4.1016
35	4.3976	4.0479
40	4.3647	4.0105
45	4.3435	3.9893
50	4.1727	3.8677

After 50 epochs, the final average loss on the test set was approximately **3.87**, corresponding to a test perplexity of about **34.53**. This indicates significant improvement in generating coherent Shakespearean-style text.

2 Task - Claim Normalization

The saved model checkpoints for this task can be accessed here

2.1 Explanation of Preprocessing Steps

The preprocessing pipeline ensures that noisy social media posts are cleaned and standardized for effective claim normalization. The main steps are:

1. **Lowercasing:** Converts all text to lowercase to reduce vocabulary size and handle case-insensitivity.
2. **Contraction Expansion:** Common contractions (e.g., *"he'll"* → *"he will"*) are expanded using `contractions.fix()`. Used `contractions` python package for this.
3. **Abbreviation Expansion:** Abbreviations are expanded using a regex-based lookup from custom `abbreviation.json`, built using data from `https://github.com/google-research-datasets/WikipediaAbbreviationData` and manual curation. The list includes 37,211 abbreviations, enabling conversion like *"Gov."* → *"Governor"*.
4. **URL Removal:** All hyperlinks are removed using a regex to eliminate irrelevant noise.
5. **Special Character Filtering:** Non-alphanumeric symbols are removed except useful punctuation (e.g., .,!?), simplifying the input.
6. **Whitespace Normalization:** Extra spaces are stripped to ensure clean tokenization.

These steps collectively produce cleaner, more uniform input for the models, improving training quality and prediction accuracy.

2.2 Model Architecture and Hyperparameters Used

We experimented with two state-of-the-art transformer-based sequence-to-sequence models: **BART** and **T5**, fine-tuning them for the task of claim normalization. The goal was to convert noisy social media posts into structured and factual claims using the CLAN dataset.

Model Variants

- **BART:** We used the `facebook/bart-base` model, which combines a bidirectional encoder and an auto-regressive decoder. BART is particularly suited for text generation tasks such as summarization and paraphrasing, making it a good fit for claim normalization.
- **T5:** We utilized `google-t5/t5-small`, a unified text-to-text transformer that frames all NLP tasks as text generation problems. It is lightweight and efficient for limited resource environments like Google Colab and Kaggle.

Tokenization and Sequence Length

- **Input Length:** Each social media post was tokenized with a maximum length of 512 tokens.
- **Output Length:** The normalized claim was tokenized with a maximum length of 128 tokens.
- Both input and output sequences were padded and truncated as necessary to ensure uniformity across batches.

Hyperparameters Used

- **Optimizer and Learning Rate:** The models were trained using the AdamW optimizer with a learning rate of 2×10^{-5}.
- **Batch Size:** The training batch size was set to 8, while the evaluation batch size was set to 4.
- **Epochs:** Each model was trained for 10 epochs, which provided a good trade-off between performance and compute time on constrained hardware.
- **Evaluation Strategy:** Evaluation was performed at the end of each epoch using validation loss.

2.3 Model Training and Evaluation Results

Training and Validation Loss Tables

Epoch	Train	Val
1	3.3850	1.3182
2	0.8265	0.5388
3	0.4832	0.5025
4	0.4195	0.4935
5	0.3811	0.4921
6	0.3428	0.4884
7	0.3314	0.4886
8	0.3363	0.4880
9	0.3218	0.4908
10	0.3114	0.4910

Figure 2: **BART Loss per Epoch**

Epoch	Train	Val
1	4.3164	1.2377
2	1.4354	0.9096
3	0.9259	0.8174
4	0.8389	0.7654
5	0.7771	0.7332
6	0.7294	0.7130
7	0.7144	0.6979
8	0.7263	0.6895
9	0.7120	0.6844
10	0.7113	0.6835

Figure 3: **T5 Loss per Epoch**

Evaluation Metrics (Validation Set)

Metric	BART	T5
ROUGE-L Precision	0.5153	0.4056
ROUGE-L Recall	0.5095	0.3592
ROUGE-L F1	0.5116	0.3793
BLEU-4	0.2975	0.1641
BERTScore Precision	0.8941	0.8461
BERTScore Recall	0.9033	0.8598
BERTScore F1	0.8986	0.8528

Table 3: Evaluation Metrics on the Validation Set for BART and T5

Training and Validation Loss Plots

Figure 4: BART Train Loss Plot

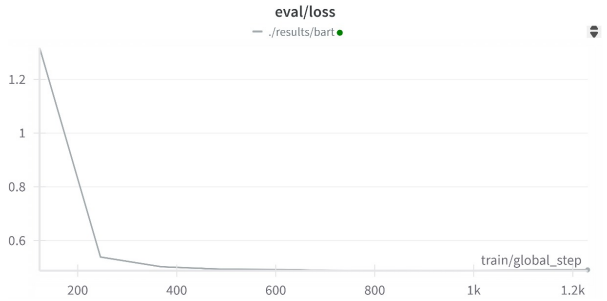

Figure 5: BART Val Loss Plot

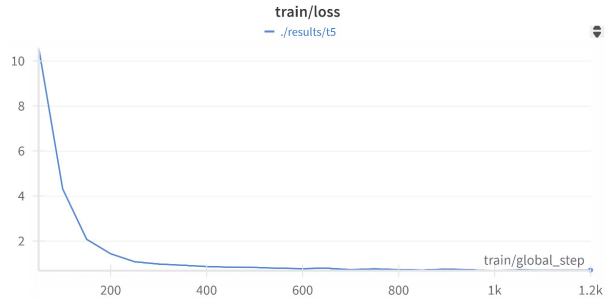

Figure 6: T5 Train Loss Plot

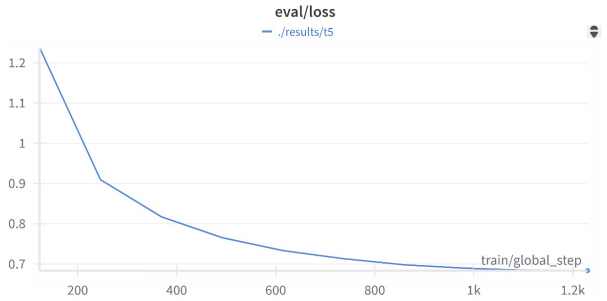

Figure 7: T5 Val Loss Plot

2.4 Evaluation on Test Set

Metric	BART	T5
ROUGE-L Precision	0.6217	0.4193
ROUGE-L Recall	0.6175	0.3741
ROUGE-L F1	0.6188	0.3940
BLEU-4	0.3930	0.1711
BERTScore Precision	0.9178	0.8582
BERTScore Recall	0.9249	0.8700
BERTScore F1	0.9213	0.8639

Table 4: Test Set Evaluation Metrics for BART and T5

2.5 Comparative Analysis of Model Performance

The evaluation results on both the validation and test sets indicate a clear performance difference between the two models.

Validation Set Analysis:

- **BART** achieved lower training and validation losses compared to T5, suggesting faster convergence and better generalization on the validation set.
- The evaluation metrics on the validation set, including ROUGE-L, BLEU-4, and BERTScore, were consistently higher for BART.

Test Set Analysis:

- On the test set, BART outperformed T5 by a significant margin. The ROUGE-L and BLEU-4 scores for BART were substantially higher, indicating more accurate and coherent claim normalization.
- BERTScore results further support this observation, with BART obtaining superior precision, recall, and F1 scores compared to T5, which demonstrates its ability to capture semantic similarities more effectively.

Overall Comparison:

- **BART** shows robust performance across all metrics, making it the preferred model for the claim normalization task in this study.
- **T5**, while effective, lags behind BART in both lexical and semantic evaluations. This might be due to its smaller model size or the inherent differences in the architecture, which could affect its capacity to handle the nuances in noisy social media text.

2.6 Discussion on Resource Constraints and Model Selection

Due to limited GPU resources on platforms like Google Colab and Kaggle Notebooks, our model selection and training configuration were carefully designed to balance performance with computational efficiency. Key considerations included:

- **Model Complexity:** We selected lightweight variants such as `bart-base` and `t5-small` to reduce the memory footprint and computational overhead while still leveraging state-of-the-art performance.
- **Batch Size and Sequence Length:** A smaller batch size (8 for training, 4 for evaluation) and moderate sequence lengths (512 tokens for input and 128 for output) were used to avoid exceeding GPU memory limits and to ensure smooth training.
- **Training Duration:** Limiting the number of training epochs (10 in our experiments) helped to reduce the overall training time, which is critical when computational resources are constrained.
- **Efficient Checkpointing:** Saving only the best-performing model and using early stopping strategies minimized both disk usage and training time, while still capturing the model's optimal performance.

Overall, the choice of lightweight model variants and efficient training configuration enabled us to fine-tune both BART and T5 within the computational limits of a typical Colab or Kaggle notebook.

3 Task - Multimodal Sarcasm Explanation (MuSE)

The saved model checkpoints for this task can be accessed here

3.1 Data Preprocessing

The preprocessing pipeline for the MuSE task integrates multimodal information through the following steps:

1. **Data Loading:** Image descriptions and detected objects are loaded from preprocessed pickle files containing annotations for each image.
2. **Object Filtering:** Detected objects are filtered using a confidence threshold (0.6). Only high-confidence labels are retained and concatenated.
3. **Modality Fusion:** The image caption, description, object labels, and sarcasm target are concatenated into a single `combined_input` string to capture rich context.
4. **Tokenization:** Inputs and sarcasm explanations are tokenized using the BART tokenizer with padding and truncation (`max_inp_length`, `max_output_length`).
5. **Image Processing:** Images are loaded via PIL and converted to RGB format for ViT processing.
6. **Dataset Structure:** A custom `MOREDataset` class organizes the tokenized inputs, images, and targets, with a custom collate function for batch assembly.

This pipeline ensures that both visual and textual data are seamlessly integrated for effective multimodal learning.

3.2 Model Architecture and Hyperparameter Settings

Model Overview:
The proposed model, **TURBOModel**, is a multimodal architecture designed to generate textual explanations by fusing visual and textual inputs. It leverages two pretrained transformer-based backbones:

- **ViT (Vision Transformer)**: Extracts high-level semantic features from input images using the `google/vit-base-patch16-224-in21k` model.
- **BART (Bidirectional and Auto-Regressive Transformers)**: Encodes text and performs conditional generation, utilizing the `facebook/bart-base` checkpoint.

Fusion Mechanism:
To effectively merge the visual and textual modalities, the model incorporates a gated fusion mechanism enhanced with multi-head attention modules. The process is as follows:

1. **Feature Projection:** Image features extracted from ViT are projected into a shared embedding space (fusion dimension of 768) via a linear layer.
2. **Multi-Head Attention:**
   - **Text Attention:** A multi-head attention module (8 heads) is applied to the text encoder outputs.
   - **Image Attention:** A similar attention module (8 heads) is applied to the projected image features.

3. **Gated Fusion:** The attended image and text features are combined with the original embeddings using two gating functions. Let I denote the image features and T the text embeddings. Define:

$$F_{vt} = \text{Attention}(T) \odot I,$$
$$F_{tv} = \text{Attention}(I) \odot T.$$

Then, the following fusion functions are computed:

$$F_1 = \sigma(W_i \cdot I + b_i) \odot F_{tv} + \Big(1 - \sigma(W_i \cdot I + b_i)\Big) \odot F_{vt},$$
$$F_2 = \sigma(W_t \cdot T + b_t) \odot F_{tv} + \Big(1 - \sigma(W_t \cdot T + b_t)\Big) \odot F_{vt},$$
$$F_3 = \sigma(W_i \cdot I + b_i) \odot I + \Big(1 - \sigma(W_i \cdot I + b_i)\Big) \odot F_{tv},$$
$$F_4 = \sigma(W_t \cdot T + b_t) \odot T + \Big(1 - \sigma(W_t \cdot T + b_t)\Big) \odot F_{vt},$$

where σ is the sigmoid function and W_i, W_t are learnable parameters.

4. **Final Representation:** The final fused representation is computed as:

$$Z = T + \sum_{i=1}^{4} \alpha_i F_i$$

and is subsequently provided as the encoder output to the BART decoder for generating the final explanation.

Table 5: Hyperparameters Used

Hyperparameter	**Value**
Batch Size	16
Optimizer	AdamW
Learning Rate	1×10^{-5}
Epochs	20
Max Input Length	256 tokens
Max Output Length	128 tokens
Fusion Dimension	768
Attention Heads	8 (image and text)
Beam Search	5 beams

Loss Function:
The model uses cross-entropy loss computed on the BART decoder outputs during training, with label padding tokens ignored via the built-in masking mechanism.

Training Details:
The training leverages PyTorch's DataLoader for efficient mini-batch processing, supporting multi-GPU setups when available. Model checkpoints are saved per epoch based on the lowest validation loss, ensuring the best performing model is retained.

3.3 Training and Validation Loss per Epoch

Table 6: Training and Validation Loss per Epoch

Epoch	Training Loss	Validation Loss
1	5.1527	1.2166
2	0.7828	0.3279
3	0.3432	0.2522
4	0.2717	0.2280
5	0.2410	0.2149
6	0.2223	0.2076
7	0.2081	0.2037
8	0.1956	0.1993
9	0.1862	0.1972
10	0.1770	0.1970
11	0.1703	0.1944
12	0.1615	0.1942
13	0.1561	0.1952
14	0.1483	0.1943
15	0.1409	0.1941
16	0.1343	0.1953
17	0.1275	0.1968
18	0.1226	0.1983
19	0.1149	0.2017
20	0.1072	0.2037

3.4 Evaluation Metrics per Epoch

Table 7: Evaluation Metrics per Epoch

Epoch	Val Loss	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	BERT P	BERT R	BERT F1
1	1.2166	0.4732	0.3041	0.4432	0.4236	0.3438	0.2877	0.2457	0.3496	0.9059	0.8996	0.9025
2	0.3279	0.4839	0.3213	0.4568	0.4419	0.3621	0.3061	0.2620	0.3759	0.9103	0.9014	0.9055
3	0.2522	0.5050	0.3448	0.4789	0.4520	0.3758	0.3216	0.2785	0.4006	0.9171	0.9066	0.9115
4	0.2280	0.5186	0.3666	0.4966	0.4581	0.3862	0.3340	0.2930	0.4212	0.9194	0.9080	0.9135
5	0.2149	0.5158	0.3616	0.4939	0.4582	0.3861	0.3341	0.2931	0.4109	0.9216	0.9096	0.9153
6	0.2076	0.5239	0.3693	0.5019	0.4616	0.3889	0.3371	0.2971	0.4211	0.9218	0.9107	0.9160
7	0.2037	0.5323	0.3762	0.5087	0.4587	0.3868	0.3369	0.2980	0.4199	0.9239	0.9118	0.9176
8	0.1993	0.5341	0.3774	0.5108	0.4723	0.4007	0.3497	0.3100	0.4326	0.9230	0.9122	0.9173
9	0.1972	0.5313	0.3733	0.5106	0.4633	0.3902	0.3398	0.3008	0.4222	0.9234	0.9115	0.9172
10	0.1970	0.5394	0.3837	0.5160	0.4723	0.3988	0.3477	0.3077	0.4320	0.9241	0.9134	0.9185
11	0.1944	0.5343	0.3752	0.5086	0.4643	0.3887	0.3359	0.2951	0.4245	0.9238	0.9115	0.9174
12	0.1942	0.5483	0.3900	0.5213	0.4767	0.4043	0.3544	0.3153	0.4372	0.9263	0.9145	0.9201
13	0.1952	0.5541	0.3915	0.5253	0.4818	0.4063	0.3543	0.3122	0.4430	0.9249	0.9140	0.9192
14	0.1943	0.5600	0.4041	0.5337	0.4883	0.4150	0.3631	0.3220	0.4538	0.9272	0.9156	0.9212
15	0.1941	0.5529	0.3906	0.5281	0.4818	0.4070	0.3545	0.3127	0.4401	0.9267	0.9152	0.9207
16	0.1953	0.5562	0.3954	0.5280	0.4809	0.4063	0.3535	0.3107	0.4479	0.9258	0.9148	0.9200
17	0.1968	0.5513	0.3843	0.5225	0.4846	0.4072	0.3547	0.3120	0.4401	0.9239	0.9149	0.9192
18	0.1983	0.5727	0.4091	0.5459	0.4939	0.4186	0.3651	0.3225	0.4538	0.9280	0.9168	0.9222
19	0.2017	0.5672	0.4038	0.5387	0.4941	0.4195	0.3680	0.3266	0.4570	0.9256	0.9162	0.9207
20	0.2037	0.5664	0.3972	0.5380	0.4884	0.4109	0.3589	0.3177	0.4468	0.9275	0.9169	0.9219

3.5 Sample Generated Sarcasm Explanations on the Validation Set (Last Epoch)

Table 8: Examples of Generated Sarcasm Explanations

Prediction	**Reference**
the author is pissed at <user> for such awful network in malad.	the author is pissed at <user> for not getting network in malad.
the author hates waiting for an hour on the tarmac for a gate to come open	nothing worst than waiting for an hour on the tarmac for a gate to come open in snowy, windy chicago.
spring isn't the best thing about spring.	nobody likes getting one hour of their life sucked away.
having a salivary gland biopsy on monday morning isn't a good way	having a salivary gland biopsy on monday morning is not a good way to start the new week.
it's going to be freezing weather this w-end, the high on saturday	the author is worried that the weekend is going to be freezing with a high of -1 and windchill probably -30.
the author is pissed at <user> for delaying the trains and charging everyone for each	the author is pissed that <user> keeps delaying the trains and charging everyone for each ride.
the author is pissed at <user> for invading his ts' privacy.	it's not a joyous organisation since <user>'s been monitoring the author's ts.
the quality of movies on hbo is terrible.	not quality movies on hbo now.
the author is pissed at <user> for forgetting about the private villas in euro	it isn't understandable to forget about the private villas in europe paid for by your millions.
the author is worried about the stamp collection.	bayley's worried about the stamp collection rather than the dead lizard, it isn't smart.
the author is pissed at <user> for such a bad delivery service.	the author is pissed at <user> that they said it was high priority, and ensured it'll be delivered and yet they didn't, such an awful delivery service.
jonny hasn't let himself go since retiring from rugby then.	jonny is fit and hasn't let himself go since retiring from rugby.
the author is pissed at <user> for delivering today (supposed to be yesterday)	awful service from <user>, delivering it today which was supposed to be delivered yesterday and sending this message <num> hours after delivery.
a sarcastic comment from chandler.	the canoe isn't in the water so chandler cannot paddle away.

Individual Contribution

- **Task 1:** Manan Aggarwal
- **Task 2:** Souparno Ghose
- **Task 3:** Shobhit Raj
- **Report Writing:** Everyone contributed equally

References

1. Hugging Face. (n.d.). Chapter 6: Attention mechanisms. Retrieved from http://huggingface.co/learn/nlp-course/en/chapter6/6
2. PyTorch Documentation. (n.d.). torch.nn.Embedding. Retrieved from https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html
3. Build your own Transformer from scratch using Pytorch Retrieved from https://medium.com/data-science/build-your-own-transformer-from-scratch-using-pytorch-84c850470dcb
4. Contractions Python Package. Retrieved from https://pypi.org/project/contractions/
5. Google Research Dataset for Abbreviations. Retrieved from https://github.com/google-research-dataset/WikipediaAbbreviationData/tree/main
6. Hugging Face Trainer API. Retrieved from https://huggingface.co/docs/transformers/en/main_classes/trainer
7. Facebook Bart-Base Model. Retrieved from https://huggingface.co/facebook/bart-base
8. Google T5-Small. Retrieved from https://huggingface.co/google-t5/t5-small
9. Google Vision Transformer vit-base-patch16-224. Retrieved from https://huggingface.co/google/vit-base-patch16-224
10. Palaash Goel, Dushyant Singh Chauhan, and Md Shad Akhtar. "Target-Augmented Shared Fusion-based Multimodal Sarcasm Explanation Generation." arXiv preprint arXiv:2502.07391, 2025. Available at: https://arxiv.org/abs/2502.07391