**Note for Part 1:**

Addressing the feedback we received on the base-line model, here is how we incorporated it and built the next models:

**Short form texts:**

Model: T5 (version 3)

Use Case: Quote Generation with Few-Shot Prompts

Settings:

top\_p = 0.95

temperature = 0.6

max\_length = 50

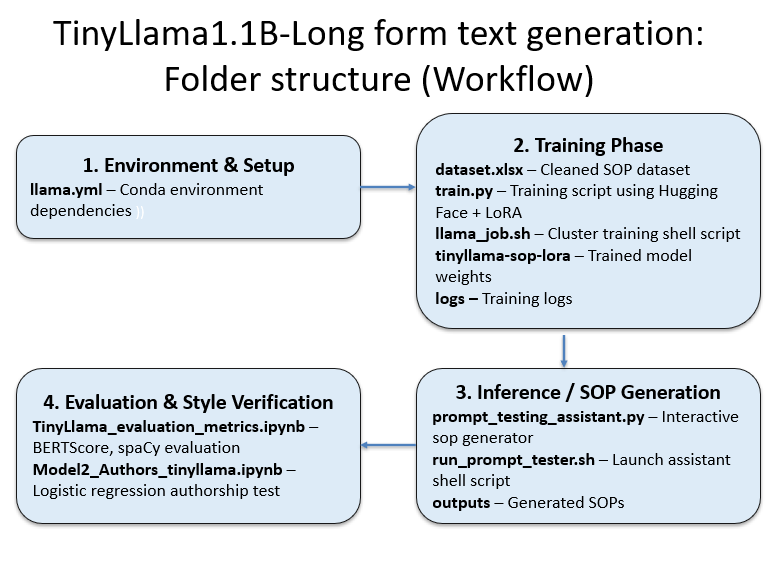
Prompting Style: Few-shot prompting with a declarative style (example-based guidance).

Special Technique: TextSETTR framework is used to condition generation based on metadata like tone, style or structure.

*Note:* Can be found in the previous submission.

**Part 2: Long form texts**

Please note that for the second phase of our project, three models were utilized (TinyLlama 1.1B, Meta Llama3.2 3B, Mistral 7B). The TinyLlama and Meta Llama models were trained on a computing cluster (instead of a notebook environment), so we have compiled these detailed notes separately to clearly document the process, settings and observations. The Mistral 7B model, however, was fine-tuned within a notebook, where explanations are embedded alongside the code.



In the second part of our project, we focused on long-form text generation based on the author’s writing style. However, we specifically chose to work with Statements of Purpose (SOPs*)* rather than multiple types like blogs, essays or letters. This decision was driven by several important considerations:

*a) Early Hallucination Issues:* Initial experiments using smaller model on a mixed dataset of SOPs, blogs, and letters led to high levels of hallucination. The model often produced content that sounded plausible but was factually incorrect, incoherent or repetitive. The model also struggled to balance the tone across diverse formats, often over-generalizing and losing specificity.

*b) More Efficient and Focused Fine-Tuning:* Training the model on a homogeneous dataset of SOPs led to more stable learning and faster convergence. Evaluation also became more meaningful, as we had consistent quality benchmarks like coherence, relevance, and clarity.

*c) Foundation for Future Expansion:* By focusing on SOPs, we established a reliable end-to-end pipeline for training, inference and evaluation. We now understand the effective token limits, optimal generation parameters, and evaluation metrics (e.g., BERTScore, cosine similarity). This positions us well to expand into other long-form formats in the future; such as blogs, essays, or academic letters, with more confidence and precision.

Hence our primary goal of this part of the project was to create a personalized Statement of Purpose (SOP) generation system using a fine-tuned language model. The system is designed to generate high-quality SOPs that mirror the writing style and structure of a dataset containing SOPs written by the author.

**Long form text generation with TinyLlama 1.1B**

**1. Data**

The source dataset was a cleaned Excel spreadsheet containing a single column titled cleaned\_text, with each entry representing a full-length SOP. The dataset was split into training and validation subsets using an 80:20 ratio.

**2. Model Selection: TinyLlama-1.1B-Chat-v1.0 and Llama 3.2 3B**

One of the most key decisions was selecting a language model that could be effectively fine-tuned on the available hardware. We initially adopted TinyLlama (1.1B), a lightweight yet capable model well-suited for experimentation. This enabled fast iterations to validate training scripts, tokenizer alignment, and the effectiveness of LoRA (Low-Rank Adaptation) strategies. Once confidence was built in the pipeline, we scaled up to Meta Llama 3B, which offered improved contextual depth and better generation fluency with author’s style. This progressive scaling avoided premature failure, allowed reuse of code, and established a controlled comparison of performance across model sizes.

To maintain compatibility and avoid errors, the model’s default tokenizer was used. The padding token was set to match the end-of-sequence (EOS) token, ensuring proper sequence termination during inference.

**3. Tokenization and Masking**

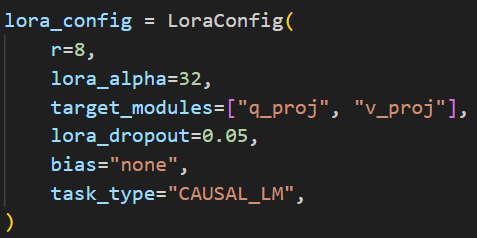
Initially, training used a 512-token sequence length to approximate the length of SOP. However, this led to frequent occurrences of NaN (Not a Number) losses. Further analysis suggested that long sequences placed excessive memory demand on the GPU, and the combination of certain attention patterns introduced instability during gradient computation. To mitigate this, the sequence length was temporarily reduced to 256 tokens, which improved training stability. Once loss values stabilized and masking strategies were integrated, the token limit was gradually increased back to 512.

To optimize training, a selective masking strategy was used during loss computation. The first reason was to avoid overfitting on repetitive, templated phrases often found at the beginning of SOPs, such as “[SOP] Statement of Purpose”, which did not contribute meaningful stylistic variation. The second reason was to ensure model focus on more expressive, author-specific content typically located in the latter half of the sequence where motivation, goals and tone become more unique.

**4. Parameter-Efficient Fine-Tuning (PEFT)**

To make model training viable on modest hardware, the project utilized PEFT through LoRA. This method inserts low-rank matrices into attention layers, allowing the model to learn new tasks by tuning only a small subset of parameters. The configuration used an LoRA rank of 8, an alpha value of 32, and a dropout rate of 0.05. These values provided a balance between learning capacity and regularization.

The LoRA layers specifically targeted q\_proj and v\_proj modules, core components of transformer attention heads. This selection ensured that expressive capacity was maximized with minimal memory footprint. Compared to full fine-tuning, this technique reduced VRAM usage, shortened training time and allowed updates to focus on the most relevant parts of the model.



**5. Training**

Training was executed over 5 epochs, AdamW optimizer with a learning rate of 1e-5 was used to ensure smooth convergence. A batch size of 2 was used to accommodate the sequences and fit within GPU memory.To prevent model corruption, runtime checks were included to skip batches that produced NaN losses. Once training concluded, the fine-tuned model and tokenizer were saved to the tinyLlama-sop-lora directory for reuse and deployment.

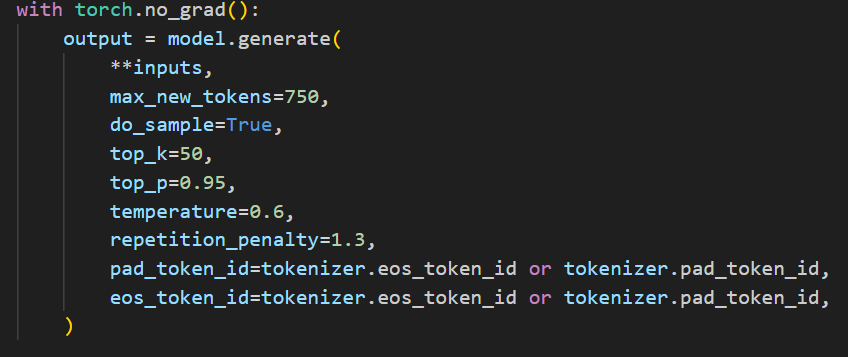
Loss reduction across epochs showed steady convergence:

This trend was smooth indicating good training.

**6. Inference:**

The generation script allows users to input any custom SOP prompt and observe how the model generates text in response. The script logged each prompt and output, supported on-demand saving, and enabled qualitative analysis of different prompt styles.

During inference several decoding parameters were carefully selected to strike a balance between fluency, coherence and creativity in the generated SOPs. The top\_k parameter was used to restrict token selection to the top 50 most probable candidates, reducing the likelihood of low-quality or erratic word choices. top\_p (nucleus sampling) was set to 0.95 to ensure that the model dynamically chose from a range of probable tokens while maintaining diversity. The temperature was set to 0.6 to smooth out the probability distribution, encouraging the model to make confident yet varied decisions in its word selection. Additionally, a repetition\_penalty of 1.3 was applied to discourage the model from generating redundant or overly repetitive phrases.



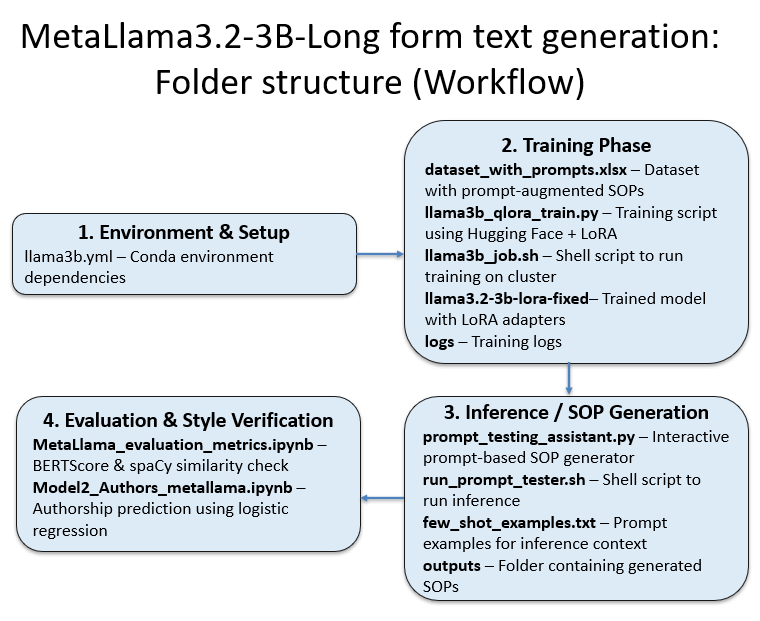
In early tests and experiments ran, common issues included prompt repetition, abrupt endings, and incomplete narratives. To mitigate this, max\_new\_tokens were increased from 300 to 600-700, enabling 500-600 word outputs. Postprocessing also removed generic letter closings to keep the focus on substantive content. While these adjustments led to noticeable improvements, it did not fully resolve the problems, hence further refinements may be necessary to consistently generate well-structured, complete documents.

**Evaluation metrics:**

NLP evaluation metrics BERTScore (F1) and spaCy Similarity were used to understand how well the model could replicate the writing style, coherence and semantic richness of long-form academic documents. Five generated SOPs were evaluated. The BERTScore F1 values ranged from 0.8137 to 0.8226, indicating strong semantic overlap with the originals. Similarly, spaCy similarity values remained consistently high, between 0.9818 and 0.9898, reflecting structural and lexical consistency. These results suggest that the model successfully learned and reproduced key patterns from the user’s SOPs in terms of both content and style.

To complement this analysis, a logistic regression model was trained on a combined authorship dataset. This dataset included the user’s original SOPs alongside a larger corpus of semi-formal long-form writing samples from various other authors. Once the model was trained, it was used to classify SOPs generated by the fine-tuned model. The goal was to test if these newly generated SOPs matched the author’s personal writing style. Four of five SOPs were predicted with high confidence (above 88%), indicating strong alignment with author’s writing style. The robotics SOP had a slightly lower confidence (58.94%) but was still attributed to the author.

**Long form text generation with Meta Llama 3.2 3B**

****

**1. Data**

Prompts were added to the dataset using the following structure,

*“Prompt: Generate a long-form academic Statement of Purpose:*

*Response:*

*STATEMENT OF PURPOSE ...”*

Prompt-based dataset was more effective than simple prefixing with tags like [SOP], as it provided clear task intent, aligned better with how users in general interact. It also helped reduce hallucinations, improve style retention and enhance generalization to varied prompt formulations during inference. An 80:20 split was done for training and validation sets.

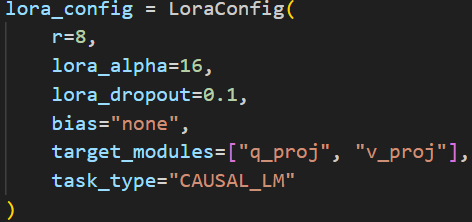
**2. Model**

The TinyLlama model allowed for fast iterations while validating training scripts, tokenizer behaviour, and LoRA integration strategies. Once the pipeline proved stable and effective, we transitioned to the Meta Llama 3B model. This transition was expected to better handle long-form SOPs, enhancing contextual depth, fluent generation and stronger retention of authorial style.

Llama 3B's ability to process longer sequences (up to 2048 tokens) and its compatibility with advanced memory optimization techniques like AMP (mixed precision) and gradient accumulation makes it a better fit. The padding tokens were masked.

Similar to the previous model, LoRA (Low-Rank Adaptation) was used for parameter-efficient fine-tuning, the target modules were q\_proj, v\_proj and the configuration is r=8, alpha=16, dropout=0.1. For TinyLlama, we increased the alpha to 32 to amplify the influence of the LoRA-adapted weights, compensating for its limited representational capacity. At the same time, we reduced the dropout to 0.05 to retain more signal during training, since the model needed more exposure to gradients to learn nuanced writing style and structure in SOPs.

For this model, we opted for the more common setup of alpha=16 and dropout=0.1, as it already has strong generalization capability and doesn’t likely require aggressive scaling of the LoRA adapter outputs. These settings were iteratively tested and chosen based on observed loss stability and quality of generated outputs.



**3.Training:**

The training script for Llama 3B was optimized for long-form learning as follows:

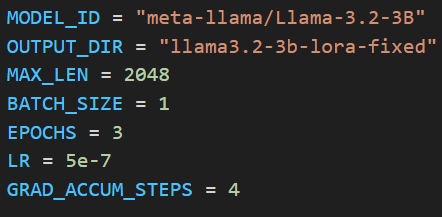
Sequence Length: 2048 tokens (full SOP coverage)

Optimizer: AdamW, LR = 5e-7

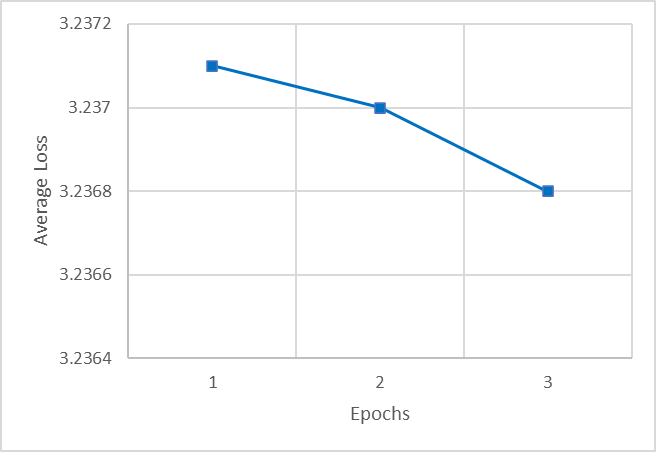
Batch Size: 1 (compensated with Gradient Accumulation =4)

Mixed Precision: Enabled via AMP

Clamping Token IDs: Prevented out-of-bound errors due to vocab drift



There were no major spikes in the validation loss despite the sequence length and large vocab, the training remained stable.



**4. Inference:**

The *prompt\_testing\_assistant.py* loads the trained model and allows interactive SOP generation:

Few-shot Prompting: Reads from few\_shot\_examples.txt (This essentially reinforces the structural expectations, improve consistency, maintains alignment between training and inference and for a stable long form generation).

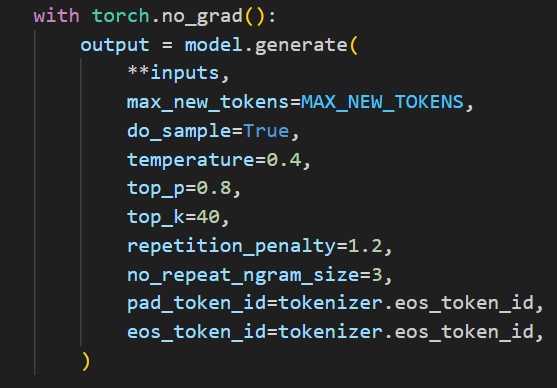
Max Tokens: 1500

Decoding Settings:

temperature = 0.4 (conservative and fluent)

top\_p = 0.8, top\_k = 40 (token diversity control)

repetition\_penalty = 1.2, no\_repeat\_ngram\_size = 3



During inference, both few-shot prompts and user-provided prompts were used together to guide text generation. The few-shot prompts, served as in-context examples that demonstrated how a well-structured Statement of Purpose (SOP) should be written. These examples helped the model internalize patterns related to structure, tone, and coherence across SOPs. In addition to these demonstrations, a user-specific prompt was also provided at runtime. This prompt described the topic or academic goal the user wanted the SOP to focus on, such as a desire to study artificial intelligence for health applications. Together, these inputs were concatenated and fed into the model, with the few-shot examples offering stylistic and structural guidance, and the user prompt ensuring topical relevance. This two-layer prompting strategy leveraged the model's ability to generalize from examples while still tailoring its output to the specific context provided by the user, ultimately resulting in more coherent, personalized, and contextually appropriate SOPs.

**5. Observations:**

The model consistently generated long, coherent SOPs with an average output length of 400–600 words. It demonstrated strong stylistic mimicry, tone and structure of the author’s original dataset. Prompt-based supervision also contributed to better alignment between training and inference behaviours.

Despite these strengths, word repetition, slightly weak conclusion sections were noted in some outputs. These issues suggest potential areas for refinement, such as incorporating structured prompt templates or applying postprocessing to reinforce strong closings. Essentially, the system performed reliably, the outputs generated with this model, showed greater stylistic retention and narrative coherence. Longer, more fluent outputs were achieved, resolving early cut-offs observed in smaller model. Also, the prompt embedded training also helped reduce hallucination.

**6. Challenges:**

Throughout the development and training process, several technical and data related challenges emerged. These were addressed as follows:

1. NaN loss during training: Early training runs encountered NaN losses due to token IDs exceeding the model’s vocabulary. To resolve this, token IDs were clamped to stay within 0 to vocab\_size - 1, and out-of-bound values were replaced with pad\_token\_id. This eliminated NaNs and ensured stable training.
2. RoPE Scaling validation error: Loading the model with certain rope\_scaling fields caused configuration errors. This was fixed by rewriting or patching the LlamaConfig dictionary to include only valid keys (e.g., name and factor), enabling smooth adapter loading.
3. Short output length (~250 words): The model often stopped early despite high max\_new\_tokens. Retry logic was added to enforce a minimum word count, while decoding was refined using min\_new\_tokens, repetition\_penalty, and no\_repeat\_ngram\_size. This led to better complete and fluent generations.
4. Hallucination and overgeneration: Without strict decoding, some outputs produced repetitive or incoherent text. Applying controlled decoding strategies (top\_k, top\_p, temperature, and eos\_token\_id) improved output structure and relevance.
5. Model compatibility and inference speed: Training long sequences with a large vocabulary strained GPU memory. Mixed precision (AMP), gradient accumulation and efficient checkpointing helped balance performance.

**Evaluation metrics:**

Compared to TinyLlama, the MetaLlama model demonstrated slightly improved evaluation results on both BERTScore and spaCy similarity metrics. While TinyLlama achieved BERTScore F1 values between 0.8137 and 0.8226, the second model showed a slightly higher range of 0.8136 to 0.8232. Similarly, the spaCy similarity scores for MetaLlama were consistently high, ranging from 0.9756 to 0.9907, marginally surpassing TinyLlama’s 0.9818 to 0.9898 range. These improvements, although moderate, suggest that MetaLlama had a stronger grasp of both semantic overlap and lexical structure. The prompt-based training, extended context window and improved stylistic retention likely contributed for better reproducing the author's tone and long-form structure with greater fluency and depth.

The authorship classification model, predicted the generated SOPs as being authored by the author with confidence scores ranging from 56.59% to 88.80%, indicating generally strong stylistic alignment, though with slightly more variability than TinyLlama (mostly scored above 88%).

**Conclusion:**

In this phase of our project, we successfully developed a long-form SOP generation system using TinyLlama and Meta Llama models. The second model, in particular, demonstrated strong capabilities in capturing the author's unique writing style, producing outputs that were well-structured, context-aware, and semantically rich. Compared to the earlier TinyLlama phase, this approach offered deeper contextual modelling, improved stylistic fidelity and more efficient inference.

To build on these results and further enhance generation quality, consistency, robustness across varied prompts and better handling of nuanced writing task, we introduced a larger model, Mistral 7B. These developments collectively established a scalable foundation for future expansion into additional long-form writing formats like academic letters, blogs and essays.