# **Project Reflection: Dialogue Summarization MVP**

**Project:** AI-Powered Dialogue Summarization using Transformer Models  
 **Author:** Sri Bailoor  
 **Date:** November 2025

## **Accomplishments**

I'm particularly proud of several aspects of this project that demonstrate both technical depth and practical engineering skills. The **comprehensive data exploration pipeline** provided crucial insights into the SAMSum dataset—discovering the 10:1 compression ratio between dialogues and summaries, identifying vocabulary differences (35,409 vs 18,675 unique words), and analyzing speaking turn patterns informed my tokenization strategies and architecture decisions.

The **dual-model comparative approach** proved especially valuable. By implementing both a BERT-GPT2 encoder-decoder and a pure GPT-2 auto-regressive model, I gained hands-on experience with two fundamentally different sequence-to-sequence paradigms. Interestingly, the simpler GPT-2 auto-regressive approach outperformed the encoder-decoder model (ROUGE-1: 0.1900 vs 0.1419), teaching me that architectural complexity doesn't guarantee better performance with limited training time.

The **optimization techniques** significantly improved training efficiency. Mixed precision training, gradient checkpointing (30-40% memory savings), parallel data loading, and learning rate scheduling collectively provided 4-6x speedup. Most importantly, implementing validation during training taught me the critical importance of monitoring generalization performance for proper model selection and overfitting prevention.

## **Opportunity for Growth**

The biggest challenge I faced was **debugging the encoder-decoder model's initially poor performance**. The issue was multifaceted: I was using the wrong EOS token during generation (tokenizer.sep\_token\_id instead of gpt2\_tokenizer.eos\_token\_id), and critically, I wasn't validating during training, meaning I was saving models at arbitrary checkpoints rather than when they generalized best.

This experience taught me valuable lessons about **systematic debugging in deep learning**. First, check every configuration parameter carefully—especially in encoder-decoder models where encoder and decoder may have different tokenizers. Second, training loss alone is insufficient; validation loss is essential for identifying models that generalize well. Third, compare against simpler baselines early—implementing GPT-2 auto-regressive first would have caught BERT-GPT2 issues sooner through direct comparison.

I now understand the value of **incremental validation**: verify each component before adding complexity, include validation metrics from the start, and maintain simpler baseline models for comparison. This methodical approach would have saved significant debugging time.

## **Feedback Request**

I would greatly appreciate feedback on **optimizing the encoder-decoder architecture for better performance**.

**My Question:** Given that my GPT-2 auto-regressive model (ROUGE-1: 0.1900) outperformed the BERT-GPT2 encoder-decoder model (ROUGE-1: 0.1419), what architectural or training modifications should I prioritize to improve the encoder-decoder approach?

**Specific areas for discussion:**

* Should I experiment with different encoder-decoder combinations (BART, T5, or matching architectures like GPT-2→GPT-2)?
* Would increasing training epochs significantly help, or have I hit a fundamental limitation?
* Are there particularly sensitive hyperparameters for encoder-decoder models (learning rate, beam search parameters, generation penalties)?
* Could preprocessing or augmentation strategies specifically benefit encoder-decoder models?

I'm curious whether others have experienced similar performance differences between these architectures and what strategies helped bridge that gap. Understanding when encoder-decoder architectures typically excel versus pure auto-regressive models would help me determine if my results reflect implementation issues or fundamental architectural differences for dialogue summarization.