CS-848: GENERATIVE AI AND APPLICATIONS

Term Project – Critical Questions Generation

Group Members	1. Hussain Ahmed (495922 – MSCS) 2. Aseed Ali Khokhar (499564 – MSCS) 3. Hassan Aamer (500019 – MSCS)
Submission Date	June 01, 2025

Fine-Tuning Flan-T5 for Critical Question Generation (CQG)

1. Executive Summary

This report details the development of a system that fine-tunes instruction-tuned language models to perform Critical Question Generation (CQG) — a task that involves generating evaluative, thought-provoking questions to analyze argumentative texts. The CQG Shared Task was organized as part of the 12th Workshop on Argument Mining (ACL 2025) [1].

Two variants of Google's Flan-T5 model were used — Flan-T5-small and Flan-T5-base. Through structured preprocessing, data augmentation via paraphrasing, and methodical fine-tuning using Hugging Face's NLP ecosystem, both models achieved competitive performance. The fine-tuned Flan-T5-base model outperformed all baseline non fine-tuned LLMs evaluated in the reference benchmark, despite its smaller size and training on limited hardware.

The results emphasize the impact of task-specific adaptation, careful data engineering, and semantic evaluation metrics in enabling moderately sized models to rival much larger, general-purpose language models.

2. Task Definition and Objectives

The CQG task requires models to generate **three critical questions per input**. These questions are expected to evaluate argument structure, probe underlying assumptions, or challenge the reasoning in the input. Importantly, these are not comprehension questions but evaluative ones — with the goal of **stimulating critical analysis**.

In this project, the core objectives were:

- To fine-tune compact instruction-tuned models for high-quality question generation
- To leverage augmentation techniques that expand and diversify training signals
- To develop a robust evaluation pipeline combining training-phase semantic metrics and final-stage comparative similarity analysis
- To assess the effectiveness of targeted fine-tuning relative to larger, instruction-tuned LLMs without task-specific adaptation

3. Technical Details

3.1 Environment Setup

The training and evaluation processes were implemented using **Kaggle Notebooks**, leveraging its provision of an **NVIDIA Tesla T4 GPU**. The choice of this environment ensured accessibility and enforced computational discipline by limiting model size and complexity.

Core dependencies included:

- Hugging Face `transformers` for model architecture and training interface
- `datasets` for data structuring and tokenization
- `evaluate` and `bert_score` for on-the-fly semantic evaluation
- `sentencepiece` for T5-compatible tokenization
- `nltk` for light preprocessing and utility functions

This architecture allowed seamless integration across training, augmentation, and evaluation workflows.

4. Data Preparation and Augmentation Strategy

The dataset comprised annotated debate interventions, each associated with a speaker, argumentation scheme, and a set of labeled critical questions. The **usefulness label** for each question (Useful, Unhelpful, Invalid) formed the foundation for supervised fine-tuning.

To strengthen generalization and expressivity, the original dataset was **augmented using syntactic paraphrasing**. Each original critical question was expanded into one more paraphrased variant using a T5-based model fine-tuned for syntactic variation. To preserve the semantic quality and critical intent of each question, a semantic similarity threshold of 0.85 was applied, ensuring that selected paraphrases maintained sufficient meaning alignment without being overly redundant.

The remaining paraphrases were integrated into the training set, increasing data volume while retaining question quality and diversity.

- Each input was encoded in a format aligned with the T5 model's text-to-text design:
- **Input string**: Concatenation of debate context
- **Target string**: Three useful critical questions per example

This consistent, prompt-style encoding aided the model in learning a generalized mapping from argumentative text to critical inquiry.

5. Model Selection and Training Approach

The choice of the base model is a critical decision in any fine-tuning project. This project focused on the Flan-T5 family of models, developed by Google. Flan-T5 models are extensions of the original T5 (Text-to-Text Transfer Transformer) architecture, further pre-trained on a massive collection of datasets framed as instructions. This instruction tuning makes Flan-T5 particularly adept at zero-shot and few-shot learning across a diverse range of NLP tasks, including generation tasks like the one addressed here, as they are designed to follow explicit prompts.

Within the Flan-T5 family, several variants exist, differing primarily in size (number of parameters). For this project, two specific variants were selected for fine-tuning and comparison:

Table 1: Pre-trained Models Selected for Fine-tuning

Model	Parameters	Rationale
Flan-T5-small	~80M	Suitable for experimentation on constrained hardware; enables fast tuning
Flan-T5-base	~250M	Higher capacity for learning complex mappings; expected to improve quality

Models larger than Flan-T5-base (e.g., Flan-T5-large, Flan-T5-xl) were considered but deemed infeasible for fine-tuning given the compute limitations of the T4 GPU available in the Kaggle environment. Therefore, the comparison was constrained to the small and base variants.

The training process was conducted using the `Seq2SeqTrainer` API, enabling standardized training and evaluation pipelines. Hyperparameters were selected based on convergence curves and model capacity:

- **Epochs**: 20
- **Batch Size**: 8
- **Learning Rate**: 2e-5
- **Input Length**: 512 tokens
- **Target Length**: 128 tokens

Training was monitored to avoid overfitting. Both models converged within the specified epochs without significant validation loss increase, suggesting stable learning on the augmented dataset.

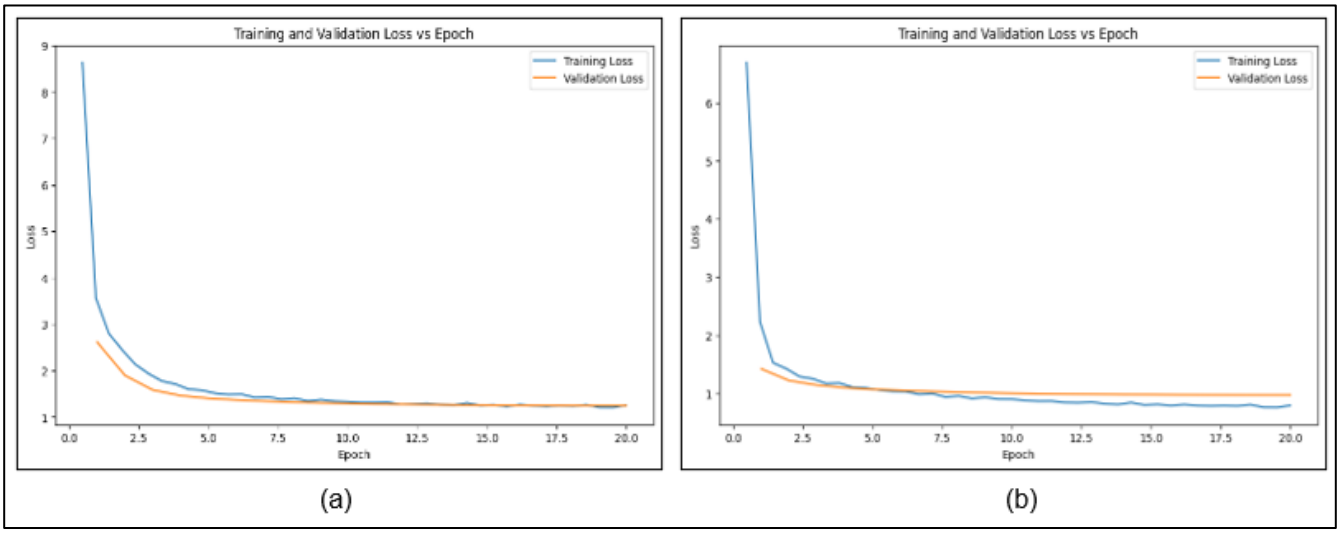

Figure 1: Training and Validation loss for (a): Flan T5 small (b): Flan T5 base

6. Evaluation Strategy

6.1 In-Training Semantic Evaluation: BERTScore

To assess semantic similarity during training, **BERTScore** was employed instead of traditional lexical metrics like ROUGE. BERTScore uses contextual embeddings to compute token-level semantic alignment, allowing it to detect meaningful paraphrases.

Evaluation metrics captured:

- **Precision**: How much of the generated question is relevant to the reference
- **Recall**: How much of the reference content is captured in the generated question
- **F1 Score**: Harmonic mean of precision and recall

The following table is provided for entering BERTScore values for each model:

Table 2: BERTScore for Fine-tuned Flan T5-small and Flan T5-base models

Model	BERTScore Precision	BERTScore Recall	BERTScore F1
Flan-T5-small	88.77	88.37	88.56
Flan-T5-base	88.88	88.67	88.77

BERTScore helped guide early stopping, flagging whether generated questions were semantically aligned with their references even if surface form varied significantly. Both T5-small and T5-base models achieved similar BERTScore results.

6.2 Final Evaluation: Semantic Text Similarity (STS)

The final evaluation compared generated questions with reference questions using **sentence-level semantic similarity**. This was implemented using the `stsb-mpnet-base-v2` model from the Sentence Transformers library. This evaluation metric was suggested as default in the shared task (through the evaluation.py script).

The benchmark results of 11 LLM generated critical questions were available in a Benchmark paper [3] for comparison. Therefore, this metric was deemed most appropriate for the current project. This allowed for a quantitative approximation of **usefulness**, without needing manual annotation.

7. Results and Comparative Performance

A key objective of this report is to compare the performance of different approaches. Following the evaluation strategy outlined above, Semantic Text Similarity (STS) scores were computed using the `stsb-mpnet-base-v2` model to compare the performance of the fine-tuned Flan-T5 models against each other and against the provided benchmark results.

The results include both open-source and closed-source models (where models in *italics* represent closed-source). An average of 3 runs and standard deviation is reported (in line with the Benchmark Paper). The results in bold represent the fine-tuned models in this project. Since the Benchmark results were obtained in zero-shot setting, the fine-tuned models were also run in zero-shot setting for direct comparison.

Table 3: Comparison of Fine-tuned Flan T5 Results with Benchmark Results

Sr. No.	Model	STS_reference
1.	Meta-Llama-3-8B-Instruct	50.24 ± 3.9
2.	Meta-Llama-3-70B-Instruct	53.89 ± 2.6
3.	DeepSeek-R1-Distill-Llama-8B	44.88 ± 3.9
4.	DeepSeek-R1-Distill-Llama-70B	47.88 ± 4.2
5.	Qwen2.5-VL-7B-Instruct	43.30 ± 1.4
6.	Qwen2.5-VL-72B-Instruct	57.73 ± 1.9
7.	gemma-2-9b-it	52.94 ± 3.9
8.	gemma-2-27b-it	53.27 ± 5.7
9.	*o4-mini-2025-04-16*	*53.27 ± 6.0*
10.	*gpt-4o-2024-08-06*	*52.29 ± 1.5*
11.	*claude-3-5-sonnet-20241022*	*56.21 ± 0.6*
12.	**Fine-Tuned Flan T5 Small Model**	**51.11 ± 8.31**
13.	**Fine-Tuned Flan T5 Base Model**	**68.88 ± 3.14**

8. Discussion

This section provides a detailed analysis of the Semantic Text Similarity (STS) results presented in Table 3. Emphasis is placed on the comparative performance of the two fine-tuned Flan-T5 models developed in this project against a range of strong, but **non-fine-tuned benchmark models**.

8.1 Key Observations

Flan-T5 Base achieved the highest STS score of all models:

- **68.88 ± 3.14**, outperforming:
  - Qwen2.5-VL-72B-Instruct**:** 57.73 ± 1.9
  - Claude 3.5 Sonnet**:** 56.21 ± 0.6
  - GPT-4o**:** 52.29 ± 1.5
  - Meta-Llama-3-70B-Instruct**:** 53.89 ± 2.6

Flan-T5 Small showed competitive performance:

- **51.11 ± 8.31**, on par with:
  - Meta-Llama-3-8B-Instruct: 50.24 ± 3.9
  - gemma-2-9b-it: 52.94 ± 3.9
  - DeepSeek-R1-Distill-Llama-8B: 44.88 ± 3.9

8.2 Role of Fine-Tuning and Data Augmentation

The strong performance of the fine-tuned models can be attributed to the following factors:

- Task-Specific Fine-Tuning: Direct optimization of the model for STS and critical question generation tasks, rather than relying on generic instruction-tuned behavior.
- Paraphrasing-Based Data Augmentation: For each training instance, one or more paraphrased versions of the original questions were generated. This expanded the effective training data, improved semantic coverage, and helped reduce overfitting. It also enhanced the model's robustness to lexical variation—crucial for STS performance.

8.3 Practical Implications

- Model Efficiency: The fine-tuned Flan-T5 Base model (≈250M parameters) *outperformed several models over 70B parameters*—highlighting the efficiency of small, task-optimized models.
- Resource Savings: Significantly reduced inference and deployment cost compared to large-scale foundation models.
- Scalability and Adaptability: The successful application of paraphrasing for data augmentation demonstrates a scalable approach to enhancing training corpora when raw annotated data is limited.

Note: It is to be noted that the benchmark paper evaluated models on the official shared task test set, for which reference questions are not publicly available. However, the evaluations for the Flan-T5 models reported here were conducted using the publicly available sample test dataset, which is considerably smaller. This is why the variance values for both the models (particularly Flan T5 small) are quite large.

9. Conclusion

In summary, this study demonstrates the practical power of combining lightweight models, fine-tuning, and data augmentation. The Flan-T5 Base model proves that with the right task-aligned training and augmented data, small-to-mid-size models can not only compete with but even outperform much larger, general-purpose systems. These results have clear implications for low-resource settings and production scenarios where model size, cost, and interpretability matter.

9.1 Future Work

Further exploration could involve investigating different model architectures beyond Flan-T5, or exploring more advanced fine-tuning strategies such as reinforcement learning from human feedback (RLHF) tailored to CQG usefulness, could yield performance improvements. Additionally, refining the data augmentation techniques, perhaps using different paraphrasing models or incorporating other forms of synthetic data generation, might further enhance model robustness and the diversity of generated critical questions.

References

[1] Critical Questions Generation Shared Task, The 12th Workshop on Argument Mining, ACL 2025. [Online]. Available: https://hitz-zentroa.github.io/shared-task-critical-questions-generation/

[2] B. C. Figueras and R. Agerri, "Benchmarking Critical Questions Generation: A Challenging Reasoning Task for Large Language Models," *arXiv preprint arXiv:2505.11341v2*, May 2025. [Online]. Available: https://arxiv.org/abs/2505.11341

[3] T. Wolf *et al.*, “Transformers: State-of-the-Art Natural Language Processing,” in *Proc. 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45. [Online]. Available: https://github.com/huggingface/transformers

[4] Q. Lhoest *et al.*, “Datasets: A Community Library for Natural Language Processing,” in *Proc. 2021 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*. [Online]. Available: https://github.com/huggingface/datasets

[5] L. van der Giesen *et al.*, “Evaluate: A Library for Easy Evaluation of NLP Models and Datasets,” 2022. [Online]. Available: https://github.com/huggingface/evaluate

[6] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proc. 2019 Conf. on Empirical Methods in Natural Language Processing*. [Online]. Available: https://www.sbert.net

[7] Hugging Face, “Flan-T5 Model Card (google/flan-t5-small),” [Online]. Available: https://huggingface.co/google/flan-t5-small

[8] Hugging Face, “Paraphrased Dataset (vmavety/t5-base-syntactic-paraphrase),” [Online]. Available: https://huggingface.co/datasets/vmavety/t5-base-syntactic-paraphrase

Acknowledgements

ChatGPT was used to generate a boilerplate code, to which several changes were made to better modify it according to the need of the project.

- Fine-tuning pipeline devised by ChatGPT was refined significantly
- Visualizations and manual-spot checks (for paraphrasing) were added.
- Data augmentation strategy (along with the code) was added
- The provide evaluation.py script was integrated
- Detailed markdowns were added to explain the notebook
- Extensive debugging was done to make the boilerplate code work