Natural Language Processing Project - Master's Degree in Data Science

Abstract Summarization on XSum: Comparative Insights into Model Training Strategies

Robin Smith[1], Sergio Verga[1], and Babak Khalilvandian[1]

[1]Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca

June 20, 2025

Abstract

This project benchmarks different abstractive summarization approaches on the XSum dataset. The focus is on model architecture, training strategies, and evaluation (with ROUGE metrics and BERTscore). We compare a classical GRU-based Seq2Seq model with modern transformer architectures, including T5-small and the instruction-tuned FLAN-T5. Experiments span zero-shot and few-shot prompting, full fine-tuning, and parameter-efficient fine-tuning (PEFT) methods such as LoRA and Prefix Tuning. We further evaluate local LLMs deployed via Ollama, including LLaMa3.2:1b and Qwen3:8b, using length-controlled prompts. Performance is assessed using ROUGE and BERTScore metrics. Results show that FLAN-T5 achieves the best balance of lexical and semantic quality (ROUGE-1 equal to 0.34 and BERTscore equal to 0.39).

1 Introduction

Automatic text summarization is a foundational task in Natural Language Processing (NLP), concerned with generating concise, coherent summaries that preserve the essential content of longer documents. Recent advances in pre-trained language models have significantly improved the quality of abstractive summaries, driving interest in both academic research and real-world applications [1].

The XSum dataset provides a challenging benchmark for this task: it pairs BBC news articles with highly abstractive, single-sentence summaries that demand both deep semantic understanding and effective compression [2]. Unlike extractive approaches, which copy spans from the source, abstractive summarization requires the model to paraphrase and synthesize new content—often leading to factuality and coherence trade-offs [3].

In this project, we conduct a systematic comparison of summarization methods under a unified evaluation. We begin with a GRU-based sequence-to-sequence (Seq2Seq) model with attention to establish a classical baseline. We then shift to transformer-based architectures, evaluating the T5-small model in zero-shot prompting, full fine-tuning, and several parameter-efficient fine-tuning (PEFT) configurations including LoRA and Prefix Tuning.

We further explore instruction-tuned models such as FLAN-T5 and examine prompting strategies in zero-shot, one-shot, and few-shot formats. In addition, we test compact local LLMs like LLaMa3.2:1b and Qwen3:8b via the Ollama platform, using length-controlled prompts (to respect the costraint of the extreme summary).

Evaluation is conducted using both surface-level metrics (ROUGE-1, ROUGE-2, ROUGE-L) and semantic metrics (BERTScore) to capture different aspects of summary quality. Our results highlight the trade-offs between architectural complexity, supervision level, and generative fidelity—offering practical insights for selecting summarization strategies across computational budgets and resource constraints.

In previous works by Shen et al. different models have been evaluated, as shown in Table 1.

Table 1: Performance Comparison of Models: ROUGE F1-scores, METEOR, and BERTScore on the XSum dataset (done by Shen et al.).

Model Name	R-1	R-2	R-L	METEOR	BERTScore
BART	0.27	0.07	0.21	0.57	0.22
FLAN-T5	0.35	0.13	0.27	**0.61**	**0.30**
LLaMA-3-8B	0.37	0.15	0.29	0.56	0.27
Gemma-7B	**0.39**	**0.18**	**0.32**	**0.61**	**0.30**

2 Dataset Exploration

The dataset used in this project is XSum (Extreme Summarization), a benchmark corpus introduced by Narayan et al. [4]. It comprises BBC news articles paired with single-sentence abstractive summaries. Designed to test the model's ability to produce concise and informative outputs, XSum poses significant challenges due to its high compression ratio and abstractive nature. The full dataset is split into training, validation, and test subsets with the following sizes:

- **Training**: 204,045 samples
- **Validation**: 11,332 samples
- **Test**: 11,334 samples

To better understand the data distribution, we analyzed the length of both documents and reference summaries. The word-level statistics for each are summarized below.

Statistic	Value
Document length (words)	
Min	23
Max	4,189
Mean	431.07
Median	395
Summary length (words)	
Min	1
Max	94
Mean	23.13
Median	22

Documents tend to be long-form text (typical news articles), while the summaries are highly compressed and always consist of a single sentence. This property supports our decision to experiment with length-controlled generation and evaluation strategies in downstream tasks. In the next figures, the distribution in terms of word count for the documents and the summaries are shown.
By computing the *Summary to Document Length Ratio* (SDLR) it is possible to observe that there are observations for which the SDLR

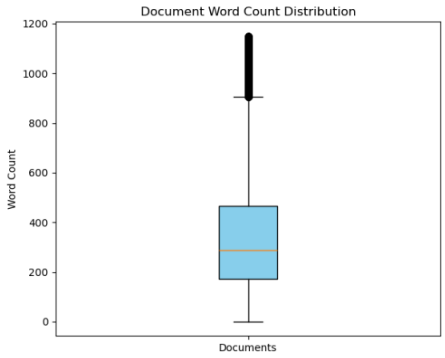

Figure 1: Word length boxplot for the documents (in the entire dataset)

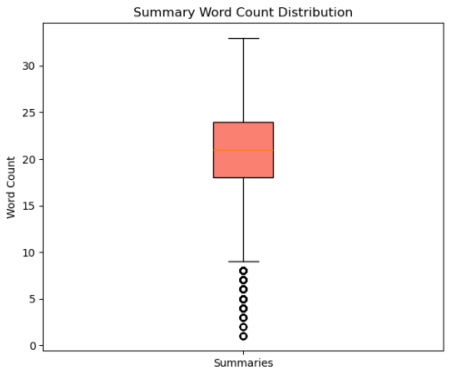

Figure 2: Word length boxplot for the summaries (in the entire dataset)

is greater or equal to 1. It means that the summary is as long, or even longer than the report, as shown in the next figure.

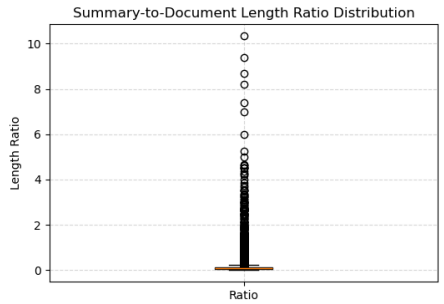

Figure 3: Summary to Document length ratio distribution

By manual inspection of such cases, it was observed that they correspond to errors in the building of the dataset and are therefore excluded with a rule (if such ratio is greater or equal to 1, the observation is filtered).

3 Experiments

To evaluate the effectiveness of various summarization approaches, we conducted a series of experiments spanning traditional sequence-to-sequence models, large pre-trained language models, and fine-tuning techniques. Our goal was to compare both classical and modern paradigms across different levels of supervision and customization.

We began with a baseline using a Seq2Seq model with attention, followed by zero-shot and few-shot prompting with base models like T5 and Flan-T5. Finally, we explored fine-tuning methods—including parameter-efficient strategies—to assess how task-specific training affects performance. Each setup was assessed using ROUGE metrics (as described at the end) and BERTscore.

3.1 Seq2Seq Baseline Model

As a baseline, we implemented a GRU-based encoder-decoder architecture enhanced with attention. This model was trained on the XSum dataset using teacher forcing and evaluated using different decoding strategies such as greedy search and beam search. The attention mechanism enables the decoder to dynamically focus on relevant parts of the input sequence during generation, thereby improving informativeness and coherence in the output.

The encoder is a unidirectional GRU that processes the tokenized input document and outputs a sequence of hidden states. The decoder is also a GRU, which, at each time step, receives an attention-weighted context vector computed from the encoder outputs.

Dropout was applied to both embedding and GRU layers to improve generalization. The decoder was trained using teacher forcing with cross-entropy loss, and the best model was selected using validation loss.

Inference and Decoding

At inference time, we experimented with:

- **Greedy decoding** (argmax at each step)
- **Top-k sampling** with $k = 50$
- **Top-p (nucleus) sampling** with $p = 0.9$

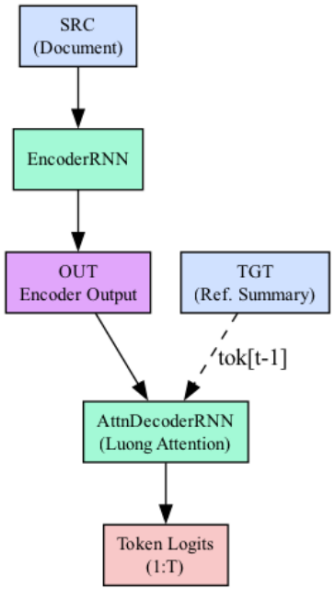

Figure 4: Encoder-Decoder GRU architecture

- **Beam search** with width 5

We also implemented repetition penalties and temperature scaling for more controlled decoding.

3.2 Base Models and Prompting Strategies

In this section, we explore various pretrained models in a zero-shot and few-shot setup. Our goal is to establish performance baselines without full fine-tuning by leveraging prompt-based learning on both local and API-based language models. This includes standard prompting as well as decoding strategy experiments.

T5-small

The `t5-small` [5] model was tested in three different inference settings using handcrafted prompts:

- **Zero-shot**: The model was directly prompted with the input document followed by an instruction such as `summarize: <document>` without any example summaries.
- **One-shot**: One example document-summary pair was included in the prompt before the new input to condition the model's generation.
- **Few-shot**: We experimented with prompting the model using 3–5 document-summary pairs concatenated before the input to encourage task-specific behavior.

These prompting modes were tested with decoding strategies such as greedy decoding, top-k sampling ($k = 50$), and nucleus sampling ($p = 0.9$). While the few-shot mode improved output consistency, the limited context window of `t5-small` constrained the number of examples that could be injected.

Google Flan-T5-Base

We evaluated the `google/flan-t5-base` model [6], which is instruction-tuned and typically outperforms vanilla T5 in zero-shot tasks. Prompts were phrased using natural instructions such as:

```
"Please generate a short summary
of the following article:"
```

We found that `flan-t5-base` generated more fluent and focused summaries than `t5-small`, particularly in zero-shot settings. Its training on diverse instruction formats makes it more robust to prompt variations.

Ollama LLMs

Two lightweight models were tested locally via the Ollama interface:

- **LLaMa 3.2:1b** [7]: A compact model with fast inference time, used with system prompts to guide generation length (e.g., `"Generate a summary of no more than 20 words:"`).
- **Qwen3:8b** [8]: A more capable model that supports detailed reasoning and longer input contexts. We evaluated its behavior using structured prompts specifying both summary intent and desired word count.

Prompting was designed with explicit control over response length and tone, based on empirical summary length distributions observed in the dataset. Both models supported instruction-following reasonably well, though Qwen3:8b produced more abstractive and diverse outputs.

3.3 Fine-Tuning T5-small

To further improve performance while minimizing computational cost, we fine-tuned the `t5-small` model on the XSum dataset. The model was trained using both full fine-tuning and parameter-efficient fine-tuning (PEFT) techniques, including LoRA and prefix tuning.

Dataset and Preprocessing

The XSum dataset was tokenized using the corresponding T5 tokenizer with a maximum input length of 2024 tokens and a maximum target length of 64 tokens. During training, padding and truncation were applied to both source documents and reference summaries. Tokenization was followed by the use of a sequence-to-sequence data collator to ensure uniform batching.

Training Configuration

We used the Hugging Face `transformers` library to load and fine-tune the pretrained `t5-small` model. The optimizer was AdamW with a learning rate of 5×10^{-5}. A linear learning rate scheduler with no warm-up steps was used. Training was performed over 10 epochs with a batch size of 2.

- **Model:** T5-small (`t5-small`)
- **Epochs:** 10
- **Learning Rate:** 5e-5
- **Batch Size:** 2
- **Loss:** Cross-entropy with teacher forcing

Parameter-Efficient Fine-Tuning (PEFT)

To reduce the number of trainable parameters and speed up training, we also explored Parameter-Efficient Fine-Tuning (PEFT) techniques beyond full fine-tuning. These methods update only a small, targeted subset of the model's parameters:

- **LoRA** (Low-Rank Adaptation) [9]: Adds trainable low-rank matrices to the attention layers, enabling efficient adaptation with minimal overhead.
- **Prefix Tuning** [10]: Optimizes a small prefix of virtual tokens prepended to the input at each layer, leaving the model weights frozen.
- **IA^3** (Infused Adapter by Scaling) [11]: Introduces multiplicative scaling vectors to key internal components (query, key, value, and MLP), requiring only minimal parameter updates.
- **Adapters** [12]: Inserts lightweight bottleneck feedforward modules between transformer layers, enabling task-specific learning while keeping the backbone frozen.

All PEFT techniques were implemented in PyTorch, with LoRA, Prefix, and IA^3 using the `peft` library, while Adapters were manually integrated to allow more flexible experimentation.

We initially explored BitFit [13], a lightweight fine-tuning approach that updates only bias terms. However, T5-small was designed without bias parameters in its main layers (e.g., Linear, LayerNorm), leaving only 512 trainable parameters in the relative attention bias. This corresponds to just 0.0008% of the model, making it unsuitable for any effective adaptation. As a result, BitFit was excluded from the final set of evaluated methods.

We selected standard hyperparameters from the respective papers and prior works:

- LoRA: rank $r = 8$, $\alpha = 32$, dropout $= 0.1$
- Prefix Tuning: 20 virtual tokens, with prefix projection
- IA^3: scaling vectors added to attention and MLP layers
- Adapters: bottleneck size $= 32$, activation $=$ ReLU

These values are commonly adopted in literature and shown to be robust across tasks. However, we acknowledge that hyperparameter tuning could further improve performance—this remains an open direction for future work.

4 Results

In this section, the results of the experiments are analyzed.

4.1 Evaluation Metrics

To evaluate the quality of the generated summaries, we adopt both lexical overlap metrics and semantic similarity measures. These include ROUGE variants and BERTScore, which together allow to evaluate both syntactic and semantic accuracy.

ROUGE-n (ROUGE-1, ROUGE-2)

ROUGE-n [14] measures the overlap of n-grams between a candidate summary and the reference summary. We report both ROUGE-1 (unigram) and ROUGE-2 (bigram) F1 scores, defined as:

$$\text{ROUGE-n}_{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Precision} = \frac{\#\text{ overlapping } n\text{-grams}}{\#\text{ n-grams in candidate}}$$

$$\text{Recall} = \frac{\#\text{ overlapping } n\text{-grams}}{\#\text{ n-grams in reference}}$$

ROUGE-L

ROUGE-L evaluates the longest common subsequence (LCS) between a candidate and reference. Let $LCS(X, Y)$ be the length of the LCS between the candidate X and reference Y:

$$P_{\text{lcs}} = \frac{LCS(X, Y)}{|X|}$$

$$R_{\text{lcs}} = \frac{LCS(X, Y)}{|Y|}$$

Then, the F1 score variant of ROUGE-L is:

$$\text{ROUGE-L}_{F1} = \frac{(1 + \beta^2) \cdot P_{\text{lcs}} \cdot R_{\text{lcs}}}{R_{\text{lcs}} + \beta^2 \cdot P_{\text{lcs}}}$$

where $\beta = 1$ balances precision and recall.

BERTScore

BERTScore [15] evaluates semantic similarity between reference and candidate summaries using contextual embeddings from a pre-trained BERT model. Let $\mathbf{x}_i$ and $\mathbf{y}_j$ be BERT embeddings of tokens from the candidate and reference, respectively:

$$\text{Precision} = \frac{1}{n} \sum_{i=1}^{n} \max_j \cos(\mathbf{x}_i, \mathbf{y}_j)$$

$$\text{Recall} = \frac{1}{m} \sum_{j=1}^{m} \max_i \cos(\mathbf{y}_j, \mathbf{x}_i)$$

$$\text{BERTScore}_{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

This metric accounts for paraphrasing and is particularly effective in evaluating abstractive summaries. To improve interpretability, BERTScore offers "rescale with baseline", which subtracts a baseline score based on unrelated sentence pairs. This normalization makes scores more consistent across sentence lengths and models, enabling fairer comparisons.

4.2 Evaluation Scores

This project compares the performance of various summarization models on the XSum

dataset using consistent evaluation metrics. The models include: (1) FLAN-T5 (`google/flan-t5-base`), (2) T5-small (pretrained and fine-tuned), (3) a GRU-based model, and (4) two local LLMs evaluated via the Ollama platform: `llama3.2:1b` and `Qwen3:8b`. The aim is to assess how architecture, prompting strategy, and training setup influence summarization quality.

FLAN-T5 was tested using three prompting strategies: zero-shot, one-shot, and few-shot. The prompts included either no examples, one demonstration, or two demonstrations respectively. T5-small was evaluated only in the zero-shot format, both in its original pretrained state and after fine-tuning on the XSum dataset. The GRU model, being RNN-based, was evaluated without prompts and tested with greedy decoding, top-k sampling, top-p sampling, and beam search.

All models were evaluated using ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore F1. Among all configurations, **FLAN-T5 in the zero-shot setting achieved the best overall performance**, with the highest BERTScore F1 (0.392381) and strong ROUGE scores. Neither one-shot nor few-shot prompting provided meaningful gains over zero-shot, demonstrating the robustness of instruction-tuning in FLAN-T5.

T5-small's pretrained version underperformed, while the fine-tuned variant improved significantly but still did not reach FLAN-T5's level. GRU-based models showed moderate performance across all decoding strategies, consistently lower than both transformer-based approaches.

These results suggest that instruction-tuned models like FLAN-T5 offer excellent summarization capabilities out-of-the-box, even without fine-tuning or in-context examples. This makes them highly practical for zero-resource and few-resource applications.

4.3 Explainability Analysis

This project analyzes how explainability reveals the behavior of the FLAN-T5 model (`google/flan-t5-base`) under zero-shot, one-shot, and few-shot prompting. Using the Inseq library, we compare how the model attributes importance to input tokens across scenarios.

FLAN-T5 is an instruction-tuned Transformer that performs tasks based on prompt structure. In zero-shot, only an instruction and the document are given. One-shot adds a single example. Few-shot includes two labeled examples. All prompts ask for a summary of an article from the XSum dataset.

We use the Input × Gradient (IxG) method for explainability. This multiplies each input token's embedding by the gradient of the model's output:

$$\text{Attribution}(x_i) = x_i \times \frac{\partial f(x)}{\partial x_i}$$

This highlights which input tokens most affect the summary generation.

In all three prompting types, the attribution consistently focuses on the main document content rather than the instruction or examples. Visual inspection confirms that the model relies primarily on the article itself when generating summaries.

This observation is supported by evaluation results, which show no significant performance difference between zero-shot, one-shot, and few-shot settings. The model's generation quality remains stable because its behavior is driven by the main input regardless of prompting style.

This project demonstrates that explainability methods like IxG can help confirm that prompting has limited impact in scenarios where the model already strongly attends to core content, offering a more precise understanding of model behavior.

5 Conclusions

This work benchmarked a variety of summarization approaches on the XSum dataset, ranging from classical Seq2Seq models to modern instruction-tuned transformers and local LLMs. Our results confirm that instruction-tuned models like `FLAN-T5` perform best out-of-the-box, achieving strong scores without task-specific training. Fine-tuning `T5-small`, particularly with PEFT techniques, yields notable gains, though still behind FLAN-T5. Local models like `Qwen3:8b` show strong semantic generation ability, as reflected in high BERTScore despite modest ROUGE, highlighting the limitations of surface-level evaluation. Overall, prompt-driven and PEFT-enhanced models offer practical, efficient summarization solutions.

6 Future Perspectives

Future work may focus on the following directions:

- scaling up fine-tuning with a larger number of epochs;
- conducting extensive hyperparameter tuning;

Table 2: Model Performance on XSUM dataset

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore F1
google/flan-t5-base Zero-shot	**0.338012**	0.118883	0.266840	0.392381
google/flan-t5-base One-shot	0.337965	**0.119797**	0.267914	**0.395298**
google/flan-t5-base Few-shot	0.337772	0.119434	**0.268080**	0.394133
llama3.2:1b (Ollama)	0.231459	0.046637	0.156299	0.228511
Qwen3:8b (Ollama)	0.219483	0.056042	0.167671	0.219449
T5-small Zero Shot	0.171081	0.022468	0.120879	0.088924
T5-small finetuned Zero-shot	0.225432	0.053187	0.174207	0.147041
GRU_pred_greedy	0.189852	0.029122	0.139330	0.135189
GRU_pred_top_k	0.147645	0.013676	0.108129	0.049020
GRU_pred_top_p	0.137238	0.012181	0.102272	0.034133
GRU_pred_beam	0.189561	0.029555	0.139431	0.138498

- incorporating a broader range of evaluation metrics;
- exploring alternative explainability methods.

In addition, human evaluation could offer more nuanced insights beyond what automated metrics can capture.

References

[1] Shakil et al., *Abstractive Text Summarization: State of the Art, Challenges, and Improvements*, 2024.

[2] Shen et al., *Evaluating LLMs and Pre-trained Models for Text Summarization Across Diverse Datasets*, 2025.

[3] Dixit et al., *Improving Factuality of Abstractive Summarization without Sacrificing Summary Quality*, 2023.

[4] Narayan et al., *Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization (XSum)*, 2018.

[5] Raffel et al., *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, JMLR 2020.

[6] Chung et al., *Scaling Instruction-Finetuned Language Models*, 2022.

[7] Touvron et al., *The Llama 3 Herd of Models*, 2024.

[8] Qwen Team, *Qwen3: Think Deeper, Act Faster*, 2025.

[9] Hu et al., *LoRA: Low-Rank Adaptation of Large Language Models*, 2021.

[10] Li and Liang, *Prefix-Tuning: Optimizing Continuous Prompts for Generation*, 2021.

[11] Liu et al., *Few-shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning*, 2022.

[12] Houlsby et al., *Parameter-Efficient Transfer Learning for NLP*, 2019.

[13] Ben Zaken et al., *BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models*, 2021.

[14] Lin, C.Y., *ROUGE: A Package for Automatic Evaluation of Summaries*, ACL Workshop on Text Summarization Branches Out, 2004.

[15] Zhang et al., *BERTScore: Evaluating Text Generation with BERT*, ICLR 2020.