Text Summarization and Small Language Model Fine tuning

Abeytunge Jayamangalage,Clervilsson Christelle

Introduction to Text Mining and NLP

March 14, 2025

1 Introduction

In the field of Natural Language Processing (NLP), summarization plays a crucial role in efficiently extracting meaningful information from large volumes of text. While state-of-the-art language models have significantly advanced performance in tasks like summarization, the computational cost and resource requirements for these models often pose a challenge, especially when operating under hardware constraints such as limited GPU memory. The goal of this project is to fine-tune a Small Language Model for summarization in a non-English language, using a GPU with a maximum of 16 GB of memory and a dataset of 5,000 documents.

This project presents several challenges: first, the computational constraints due to the limited GPU memory, which requires exploring optimization techniques to make the model training feasible. Second, there is the challenge of language selection, as a significant percentage of large language models are primarily designed for English, making it more difficult to work with non-English languages.

2 Steps

1. Dataset Selection
2. Dataset Annotation
3. Data Splitting and Model Finetuning
4. Model Evaluation

2.1 Dataset Selection

We opted for a French dataset to ensure that we could accurately interpret the results and assess their correctness. The size of the dataset posed a challenge, so we reviewed previous research on French summarization in NLP for guidance. Initially, we considered a dataset based on legal judgments called CASS. However, we ultimately selected the **OrangeSum** dataset by Eddine et al. [1], which covers articles from February 2011 to September 2020.

The dataset includes two summarization tasks: title generation and abstract generation. On average, the ground truth summaries are 11.42 words for titles and 32.12 words for abstracts, with document sizes averaging 315 and 350 words, respectively.

The "orangesum_filtered_new_spaces" dataset is a derivative of the original OrangeSum dataset. Hosted on Hugging Face, this version contains 9,020 rows and is formatted in JSON. It is designed for summarization tasks and is in French.

The primary difference between the original OrangeSum dataset and the "orangesum_filtered_new_spaces" version lies in the filtering and preprocessing applied to the latter. While the original dataset encompasses a broader range of articles, the filtered version likely includes specific preprocessing steps, such as the removal of certain articles or adjustments to whitespace, to enhance the dataset's quality for specific summarization tasks. Therefore we used a subset of this dataset for the synthetic summary generation.

2.2 Dataset Annotation

2.2.1 Data Pre-processing

As the chosen dataset ("orangesum_filtered_new_spaces") that is been filtered to some extent, as preprocessing we did remove extra spaces, normalized spaces, to ensure the cleanness.

And to save time that takes for generation, we only chose the short articles (less than 2900 characters) from the train split of the dataset and and this subset contained 5409 articles.

2.2.2 Large Language Model

To create synthetic summary of the document we had to choose a good Large Language Model that can have great performance on a french dataset and also that can be running on a limited GPU size. We compared the available models and the table 1 shows a summary of the comparison.

Model	Size	Multilingual	Strengths	Weaknesses
Jais-13B	13B	Yes (Arabic, French, English)	High-quality summaries.	Requires Colab Pro with quantization.
Qwen2.5-7B-Instruct	**7B**	Yes (English, **French**, Chinese, etc.)	**Efficient with good summarization quality**.	**LoRA/QLoRA recommended** for best performance.
Mistral-7B-Instruct	7B	Partial (mostly English, some French)	Strong performance in summarization.	LoRA recommended for optimal efficiency.
Gemma-7B	7B	Partial (mostly English, but multilingual)	Lightweight LLM from Google.	Requires quantization for optimal performance.
DeepSeek-7B	7B	Partial (some French support)	High-quality output for summarization.	Requires quantization for best results.
Nous-Hermes-2-Mistral-7B	7B	Partial (English-focused but good output)	Produces good summaries.	LoRA/QLoRA needed for efficient fine-tuning.

Table 1: Comparison of LLMs for French Summarization

Considering the above information, we used the Qwen2.5-7B-Instruct model with 4-bit quantization to stay within the memory limits of Colab's free tier for generating summaries.

However, generating a single summary took approximately one minute. To speed up the process, we attempted to use vLLM for fast inference, but it was not compatible with 4-bit quantization. Nevertheless, by optimizing our code (stopping gradient calculation, changing the computation data type to float16, etc.), we managed to reduce the generation time to 30 seconds per summary, which is still relatively slow.

2.2.3 Prompt Engineering

Without any examples, the model's results were not very good. Therefore, we decided to include a few-shot approach by adding two example summaries in the prompt to improve generation quality.

Additionally, we restricted the word limit in the prompt to 50 words to prevent overly lengthy summaries. This number was chosen considering the distribution of the original summaries provided in the dataset.

Furthermore, we restricted the `max_new_tokens` to 150 to accommodate the 50-word summary while ensuring that the generation remained concise and did not include unnecessary information.

2.2.4 Dataset Validation

Despite this limitation, the generated summaries were of good quality. We first evaluated them using **ROUGE scores**, comparing them with the **original summaries** from the OrangeSum dataset. (Table 2)

ROUGE Metric	Score
ROUGE-1	0.3098
ROUGE-2	0.1111
ROUGE-L	0.2029
ROUGE-Lsum	0.2028

Table 2: ROUGE Scores for Generated Summaries

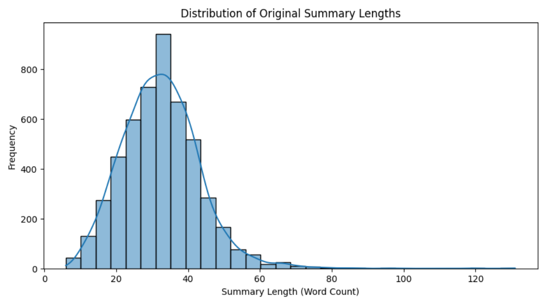
The obtained ROUGE scores indicate a reasonable level of lexical overlap between the generated summaries and the reference summaries. Specifically, the ROUGE-1 score of 0.3098 suggests that approximately 31% of unigrams are shared between the generated and reference texts, while the ROUGE-2 score of 0.1111 reflects a lower level of bigram overlap. The ROUGE-L and ROUGE-Lsum scores, 0.2029 and 0.2028, respectively, further demonstrate structural similarity in terms of longest common subsequences. While these scores suggest that the generated summaries capture key elements of the original text, ROUGE primarily measures word overlap and does not fully account for semantic equivalence.

Metric	Score
Precision (P)	0.7086
Recall (R)	0.7325
F1-score	0.7200

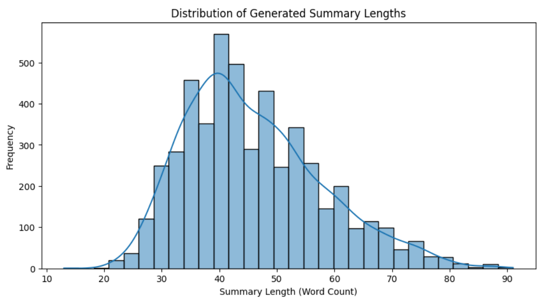
Table 3: BERTScore Evaluation of Generated Summaries

Since different summaries can express the same meaning using varying wording, we evaluated semantic similarity using BERTScore (Table 3) to determine how well the generated summaries preserve the intended meaning of the original text. BERTScore assesses similarity at the semantic level through contextual embeddings, and with an F1-score of 0.7200, the results indicate that the generated summaries maintain a strong semantic resemblance to the reference summaries.

The figures 1a and 1b show the histograms of word counts of the original summaries and the synthetic summaries.



(a) Histogram of Original Summary Lengths



(b) Comparison of Text and Summary Lengths

Figure 1: Analysis of Summarized Data

2.2.5 Post Processing

After generation, the summaries underwent several cleaning steps to ensure quality and consistency:

- Trailing spaces were removed.
- Tags used in the prompt to mark the generated summaries within the model output were stripped. These tags helped extract only the summarization portion from the output.
- Summaries that were truncated mid-sentence due to the `max_new_tokens` limit were discarded.
- Summaries containing fewer than 10 words were also removed to maintain meaningful content.

2.3 Data Splitting and Model Finetuning

We split the data into 3 parts (Training, test and validation) to avoid overfitting and ensure model generalization. To choose our Small Language Model we wanted to ensure that it was performing well in french summarization and compare some of the best before choosing our SLM and the table 4 expose the summary of our points.

Also, we had some other good examples of French-related models, but they had major issues that made us decide not to test them. For example:

- BART-FR, which requires a large training dataset. While it was strong for summarization and specialized for French, its high data requirements made it less suitable for our use case.

Model	**Size**	**Strengths**	**Fine-tuning Strategy**	**Support for Faster Training**
Qwen2.5-0.5B-Instruct	**500M**	Extremely small, fast training	Standard fine-tuning	Unsloth
Atlas-Chat-2B	2B	Good quality, multilingual	LoRA/QLoRA	Yes
mT5-Base	**580M**	Strong multilingual summarization	LoRA/QLoRA (FSDP/DeepSpeed to reduce memory)	No
Mistral-7B	7B	Best summarization quality	Use QLoRA to fit into Colab	Unsloth/LLaMA Fac

Table 4: Comparison of Different Models for Fine-tuning

- CamemBERT, which performs well for French NLP tasks but is not optimized for summarization.

When we analyzed the table, our initial choice was the mT5-Base model because it was the smallest model trained on a massive multilingual corpus. This suggested it would have high-quality French text generation capabilities. Additionally, it was trained for sequence-to-sequence tasks and performed well with limited training data. However, in our case, it did not yield satisfactory results.

Moreover, we found a research paper [2] that initially encouraged us to use this model. These papers stated that mT5, based on T5's sequence-to-sequence architecture, is well-suited for abstractive summarization, which is particularly valuable for French, where fluency and coherence in summarization are crucial. The research also showed that mT5-Base outperforms mT5-Small, suggesting that increasing model size improves performance. That is why we did not choose mT5-Small, even though it is smaller.

Our Final Choice: **Qwen 2.5-0.5B**

Due to these findings, we chose to use Qwen 2.5-0.5B.

- Qwen 2.5 incorporates advanced techniques like Dual Chunk Attention (DCA) to handle extended context lengths, supporting sequences up to 1 million tokens. This capability is particularly beneficial for summarizing lengthy French texts, ensuring coherence and relevance throughout the summary. [3]

- The Qwen 2.5-0.5B model is a causal Transformer-based language model, meaning it predicts the next word in a sequence based on previous words without looking ahead. It has 0.5 billion parameters.

Key Features:

- 24 Transformer layers with tied word embeddings (input and output embeddings share the same parameters). It has approximately 0.36B non-embedding parameters for optimized performance.

- Supports sequences up to 32,768 tokens, making it suitable for summarizing lengthy documents.

- Balances efficiency, accuracy, and scalability, making it a strong candidate for fine-tuning French text summarization.

Qwen 2.5-0.5B Instruct

We also trained a variation of this model: Qwen 2.5-0.5B-Instruct. The base model is a general-purpose model, trained to perform a wide range of tasks. However, the "Instruct" version is fine-tuned to follow instructions more accurately, such as responding precisely to prompts like "Summarize this text."

Because of this, we wanted to compare their results, so we trained both models to evaluate their performance.

2.4 Optimization

Due to the constraints of GPU size,we had to find the best way to optimize model training without compromising performance. The first element that we used was **QLoRa** (Quantized Low-Rank Adaptation), which is a technique designed to efficiently fine-tune large language models (LLMs) using significantly less GPU memory while maintaining high performance.

Moreover, in the training arguments, we added some parameters that helped us optimize the memory control.

- Fp16: Reduce VRAM usage and speed up training.
- Warmup ratio: Gradually increases the learning rate at the beginning
- gradient_accumulation_steps: Allows training with large effective batch sizes while using less VRAM

2.5 Model Evaluation

The evaluation is done for both Qwen2.5-0.5B and Qwen2.5-0.5B-Instruct models.

Before fine-tuning, we generated summaries for the test split to compare the results with the summaries generated by the fine-tuned model for the same split. We calculated the ROUGE scores and BERTScore metrics for each case.

The tables 5 & 6 show the results for Qwen2.5-0.5B, and tables 7 and 8 present the results for Qwen2.5-0.5B-Instruct.

Approach	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-Lsum**
Zero-shot	0.2032	0.0603	0.1307	0.1308
One-shot	0.1890	0.0454	0.1259	0.1260
Few-shot	0.1858	0.0466	0.1257	0.1258
Fine-tuned	**0.2623**	**0.0844**	**0.1697**	**0.1696**

Table 5: ROUGE Scores for Different Summarization Approaches (Qwen General)

A visual comparison of the performance of the two models is shown in figure 2 (ROUGE Scores) and figure 3 (BERTScore).

It is visible that the fine-tuning has improved the ROUGE scores 2, and the Qwen-General model has outperformed the Qwen-Instruct model in every case.

However, when comparing the BERTScores 3, without fine-tuning, the Qwen-Instruct model shows better results. After fine-tuning, the Qwen-General model has outperformed the Qwen-Instruct model.

Approach	**Precision (P)**	**Recall (R)**	**F1-score**
Zero-shot	0.6304	0.6910	0.6584
One-shot	0.6321	0.6744	0.6517
Few-shot	0.6336	0.6704	0.6504
Fine-tuned	**0.6928**	**0.7100**	**0.7000**

Table 6: BERTScore for Different Summarization Approaches (Qwen General)

Approach	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-Lsum**
Zero-shot	0.1861	0.0418	0.1215	0.1216
One-shot	0.1808	0.0372	0.1191	0.1189
Few-shot	0.1871	0.0370	0.1234	0.1234
Fine-tuned	**0.2178**	**0.0589**	**0.1407**	**0.1405**

Table 7: ROUGE Scores for Different Summarization Approaches (Qwen Instruct)

2.5.1 Hyperparameters tuning

We train our models multiple times while changing some parameters to find the best possible results. The main parameters that we used for hyperparameters tuning where:

- *num train epochs* : Number of passes over the dataset. Higher epochs improve learning but risk overfitting and lower values can cause underfitting
- *per device train batch size* : Larger batch sizes speed up training but require more VRAM
- *learning rate* : If it's too high the training will be unstable if it's too low the convergence will be slow
- *weight decay*: High value help reduce overfitting

For the model Qween 2.5-0.5B

Without validation

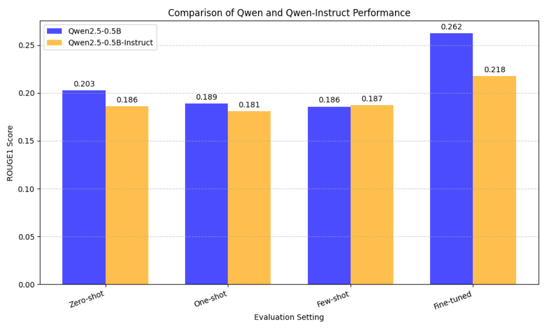
With

- learning-rate = 2e-5
- num-train-epochs = 3
- weight-decay = 0.01
- save-steps = 500
- max-steps = 500

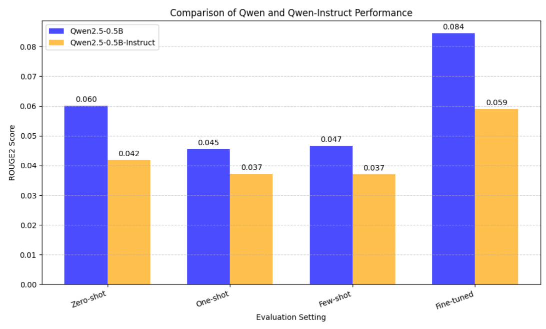
We got a loss = 1.9595 as we can see in the table 9

Approach	Precision (P)	Recall (R)	F1-score
Zero-shot	0.6549	0.6757	0.6642
One-shot	0.6562	0.6763	0.6654
Few-shot	**0.6651**	0.6765	0.6700
Fine-tuned	0.6549	**0.6951**	**0.6738**

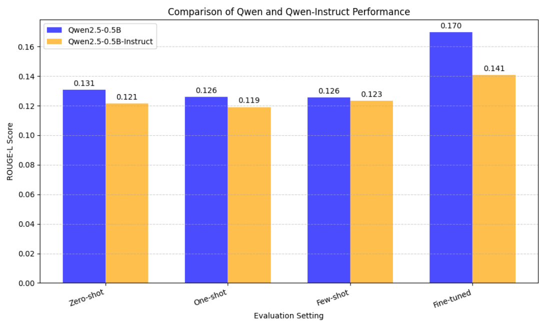
Table 8: BERTScore for Different Summarization Approaches (Qwen Instruct)



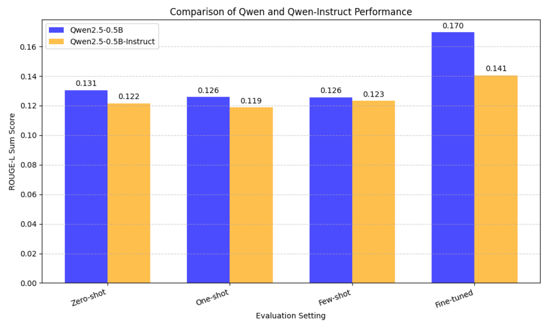
(a) ROUGE-1 Score Comparison



(b) ROUGE-2 Score Comparison



(c) ROUGE-L Score Comparison



(d) ROUGE-Lsum Score Comparison

Figure 2: Comparison of ROUGE scores for Qwen-General and Qwen-Instruct across different evaluation settings.

With validation

With

- learning rate = 2e-5
- num train epochs = 3
- weight decay=0.01
- max steps =250

We got a loss = 2.041800 as we can see in the table 11

With

- learning-rate = 5e-5

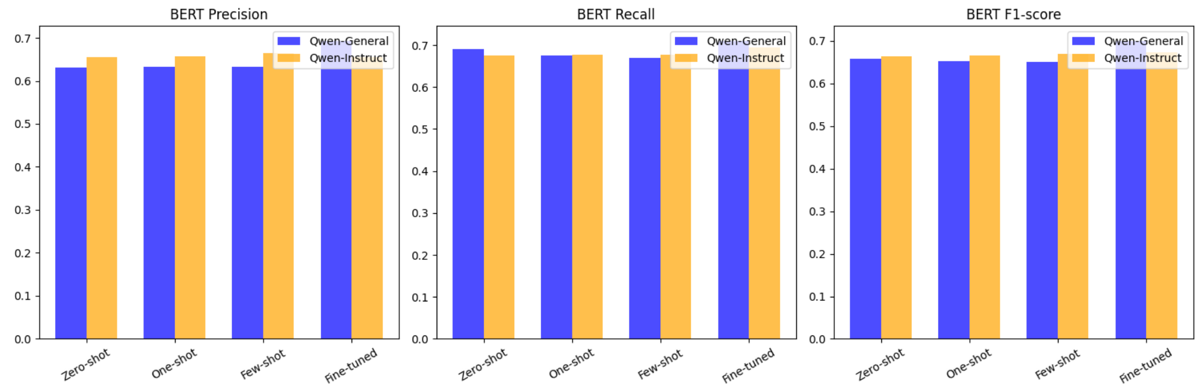


Figure 3: Comparison of BERT Precision, Recall, and F1-score for Qwen-General and Qwen-Instruct across different evaluation settings.

- num-train-epochs = 3
- weight-decay = 0.01
- max-steps =250
- per-device-train-batch-size = 8

We got a loss = 1.830000 and it was our best results 10

For the model Qween 2.5-0.5B-Instruct

We run this model only with validation

With

- learning-rate = 2e-5
- num-train-epochs = 3
- weight-decay = 0.01
- max-steps =250
- per-device-train-batch-size = 8

We got a loss = 2.131900 and it was our best results 12

Step	Training Loss
100	2.0042
500	1.9370
1000	1.8894
1500	1.9595

Table 9: Training Loss Qween 2.5-0.5B - no validation

Step	Training Loss
200	3.447700
400	2.110900
600	**1.830000**

Table 10: Training Loss Qween 2.5-0.5B - with validation

3 Conclusion

In this project, we explored the best possibilities for generating French text summarization and identified the best small language model to fine-tune for this task. We used the Qwen model with 7B parameters for summary generation and Qwen 2.5 with 0.5B parameters for fine-tuning. We trained two different versions of the same model: the base model and the Instruct model. We generated summaries using these models both before and after fine-tuning.

To evaluate the models, we used metrics such as BERT-score and ROUGE. For BERT-score, the Instruct model was consistently better before fine-tuning, but after fine-tuning, the base model outperformed it. As for the ROUGE score, the base model was always better. Additionally, we achieved the best training loss with the base model.

References

[1] Moussa Kamal Eddine, Antoine J. P. Tixier, and Michalis Vazirgiannis. *BARThez: a Skilled Pretrained French Sequence-to-Sequence Model*. 2021. arXiv: 2010.12321 [cs.CL]. URL: https://arxiv.org/abs/2010.12321.

[2] Mehrdad Farahani, Mohammad Gharachorloo, and Mohammad Manthouri. "Leveraging ParsBERT and Pretrained mT5 for Persian Abstractive Text Summarization". In: *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*. IEEE, Mar. 2021, 1–6. DOI: 10.1109/csicc52343.2021.9420563. URL: http://dx.doi.org/10.1109/CSICC52343.2021.9420563.

[3] The Pingouin. *Qwen 2.5-1M: Modèle Long Contexte*. Accessed: 2025-03-14. 2025. URL: https://www.thepingouin.com/2025/01/29/qwen2-5-1m-modele-long-contexte/.

Step	Training Loss
200	3.700600
400	2.669500
600	2.041800

Table 11: Training Loss Qween 2.5-0.5B - with validation

Step	Training Loss
200	2.967800
400	2.301100
600	2.131900

Table 12: Training Loss Qween 2.5-0.5B-Instruct - with validation