Finetuning Small Language Models for Summarization in Portuguese

Pedro Silva, Sabrina Lomelino Sartori

{pedro.silva, sabrina.lomelino-sartori}
@polytechnique.edu

March 13, 2025

1 Introduction

Large Language Models (LLMs) are the state-of-the-art in almost all natural language processing (NLP) tasks. These models are built upon the transformer architecture, which leverages the attention mechanism [1] to process and generate text efficiently.

LLMs are trained on vast corpora, often comprising nearly all publicly available text on the internet. Their training process consists of multiple phases. Initially, the model undergoes pretraining, where it is randomly initialized and then exposed to billions of tokens. This phase, which requires significant computational resources, is typically conducted by large technology companies with access to extensive computing power. During pretraining, the model learns the fundamental semantics and structure of language, enabling it to generate human-like text.

Following pretraining, these models can produce coherent and contextually relevant text. However, they often inherit biases present in their training data, potentially leading to harmful or misleading responses. To address these issues, a second phase of training is introduced, focusing on alignment with human values. This phase employs reinforcement learning techniques such as Reinforcement Learning from Human Feedback (RLHF), Reinforcement Learning from AI Feedback (RLAIF), or Direct Preference Optimization (DPO) [2, 3, 4]. These approaches refine the model by incorporating human preferences while maintaining a regularization mechanism to prevent excessive divergence from the pretrained model. The goal is to enhance the model's helpfulness, harmlessness, and honesty.

When adapting LLMs for specific tasks such as machine translation or text summarization, supervised fine-tuning (SFT) can be employed. This process mirrors pretraining but operates on a much smaller, domain-specific dataset. By fine-tuning on task-relevant examples, the model improves its ability to perform specialized language tasks effectively.

In this work, we fine-tune a small language model (SLM) for text summarization in Portuguese. To achieve this, we first curate a dataset of Wikipedia articles. Then, we leverage a pretrained large language model to generate high-quality target summaries, which serve as reference outputs for training our SLM. Finally, we evaluate the model's performance using multiple metrics, including BERTScore [5] and ROUGE [6].

By employing this structured fine-tuning approach, we aim to develop an efficient summarization model capable of producing concise and informative summaries in a foreign language.

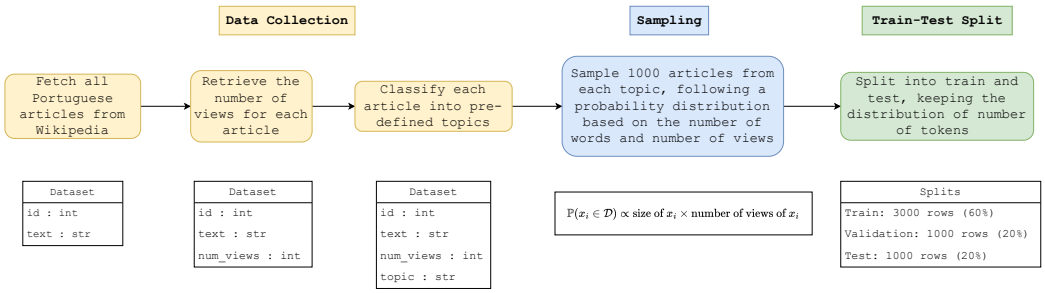


Figure 1. Workflow for dataset generation for SLM training.

2 Data Collection

A good fine-tuning dataset for summarization should consist of high-quality, well-structured texts that are both diverse and representative of the language's usage. The texts must be grammatically correct and coherent, providing clear context and logical flow, which helps the model understand how to identify the main points and condense information effectively. It's important to include documents that vary in length and complexity, ensuring the model learns to handle both short and long texts. Additionally, the content should cover a wide range of topics and styles—from formal academic or technical writing to more informal narratives—so that the model becomes robust across different domains and registers.

In this work we chosen to fetch data from **Wikipedia**, since it offers a wealth of well-curated and structured content. Articles on Wikipedia are created and continuously reviewed by a community of editors, ensuring that the language is clear, factual, and grammatically sound. This level of quality and consistency is crucial for training models, as it allows the algorithm to learn from texts that are both coherent and comprehensive. Moreover, Wikipedia spans a broad range of topics—from history and science to art and culture—providing a rich and diverse dataset that enables the model to generalize its summarization skills across different subjects. The pipeline of dataset generation for SLM training is illustrated on figure 1.

2.1 Sampling process

The Wikipedia dataset, available via HuggingFace[1], contains about one million Portuguese Wikipedia pages. For fine-tuning our model, we need to select 5,000 pages.

A random sample might include too many arbitrary or rarely seen topics, while selecting the most viewed pages ensures relevance but can lead to redundancy (for example in the top 10 most viewed pages, 4 are football-related). To balance these issues, we combine sample from different topics with a preference for top-viewed pages.

We first retrieve page view counts using the Wikimedia API and assign a topic (e.g., sports, business, culture, science) to each page with a Multi-Genre Natural Language Inference (MNLI) approach. Although a larger model like Llama could provide more nuanced topic assignments, it would be too computationally expensive.

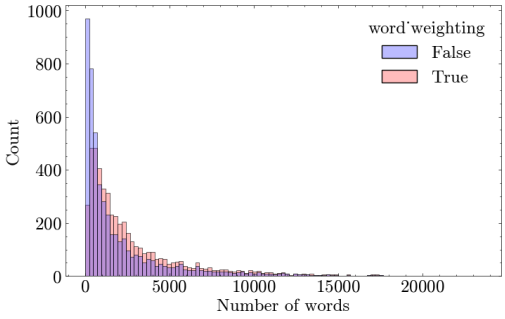
We then randomly sample 1,000 pages per topic, with the probability of selecting a page proportional to the product of its view count and word count:

$$\mathbb{P}(x_i \in \mathcal{D}) \propto (\text{number of views in } x_i) \times (\text{number of words in } x_i) \tag{1}$$

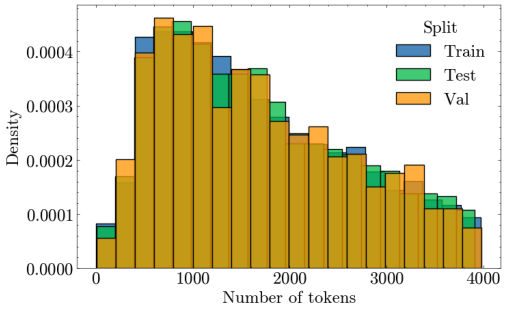
[1]The Wikipedia dataset is publicly available at `https://huggingface.co/datasets/wikimedia/wikipedia`

where $\mathcal{D}$ is the final dataset. Weighting by word count enhances the diversity of context lengths as seen on figure 2a.

Finally, to accommodate GPU memory constraints, we truncate texts exceeding 4,000 tokens at the nearest paragraph break. To ensure a balanced distribution of context lengths, we perform a stratified train-test split, allocating 3,000 samples to the training set and 1,000 samples each to the validation and test sets. The final distribution of context lengths in our dataset is illustrated at figure 2b.



(a) Effect of word weighting on the sampling process.



(b) Distribution of context length in the train/validation/test splits.

Figure 2

3 Generating target summaries

Before fine-tuning the model, we need to generate high-quality target summaries. For this, we use the `Llama-3.1-8B-Instruct`[2] model with the prompt shown on appendix B. The main reasons for choosing this model are:

- It is multilingual and supports the Portuguese language.
- It is an open-source model widely used and validated by users worldwide.
- It has been trained on a vast amount of internet data, including Wikipedia.

We generate summaries of approximately 100 words, a decision driven by the computational constraints we faced during both the fine-tuning and inference stages.

To load the model on a single 15GB GPU, we used a 4-bit quantized version. For faster inference, we leveraged the unsloth API with vLLM.

4 Finetuning the SLM

4.1 Choice of model

We have chosen the model `Qwen2.5-0.5B-Instruct`[3]. The main reasons why we selected this model were

[2] Available at `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
[3] Available at `https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`

- It is multilingual and supports Portuguese.
- It is a recent model, released in September 2024, and is recognized among the best small models.
- It has been instruction-tuned, enabling it to follow instructions effectively and generate helpful outputs.

4.2 Why do we need finetuning?

Since the model was trained on Portuguese text, we may question whether fine-tuning is truly necessary.

To evaluate this, we generated summaries using the raw model and compared them to both the original text and the target summary. This analysis uncovered some unusual behaviors, such as repetitive ideas, content in languages other than Portuguese, and information that was not present in the original text.

In the example below (translated from Portuguese), the language models were prompted to summarize the Wikipedia page of the Los Angeles Lakers basketball team. Notably, while the Llama 8B model produces a short yet concise summary, the SLM model repeatedly uses phrases like *"the team is also known"* and *"the team also had a high-quality player"*.

Page Name: Los Angeles Lakers

Llama 3.1 8B Instruct

The Los Angeles Lakers is an NBA basketball team based in Los Angeles, California. Founded in 1947 in Minneapolis, the team won five league titles before moving to Los Angeles in 1960. With stars like George Mikan, Jerry West, Wilt Chamberlain, Elgin Baylor, and Magic Johnson, the Lakers is one of the most popular franchises in the NBA, with 17 titles and 56 postseason appearances. The team is also known for its rivalry with the Boston Celtics and for having been led by coaches like John Kundla and Phil Jackson.

Qwen 0.5B Instruct

[...]
The team is also known for having a high-quality player, Elgin Baylor, who was the player with the highest average points (24.9 ppg) and rebounds (15 rpg) in his first season.
The team also had a high-quality player, Jerry West, who was the player with the highest average points (34.8 ppg) and rebounds (19.8 rpg) in his first season.
The team is also known for having a high-quality player, Rudy LaRusso, who was the player with the highest average points (61 ppg) and rebounds (10.4 rpg) in his first season.
The team also had a high-quality player, Frank Selvy, who was the player with the highest average points (14.7 ppg) and rebounds (14.7 rpg) in his first season.

4.3 Metrics

Before fine-tuning the model, it is essential to define the evaluation metrics and establish criteria for determining whether a generated text constitutes a good summary. To achieve this, we assess the generated summaries based on the following aspects:

- The text must be primarily in Portuguese.
- The length of the text should approximately match the user's requested size in the prompt.
- The **ROUGE** metric [6] is used to compare the generated summary with the target summary.
- The **BERTScore** metric [7] is applied to evaluate the semantic similarity between the generated and target summaries.

In particular, we use ROUGE-Lsum, a version of ROUGE-L adapted for summarization tasks. This metric measures the Longest Common Subsequence (LCS) between the generated and reference summaries, capturing sentence structure and fluency without requiring exact word adjacency.

The selection of both **ROUGE** and **BERTScore** as evaluation metrics is well justified. ROUGE measures the overlap of n-grams between the generated and reference texts, making it effective for assessing lexical similarity. In contrast, BERTScore leverages word embeddings to compare the semantic relationships between texts, focusing on meaning rather than exact vocabulary matches. This combination ensures a more comprehensive evaluation of both lexical accuracy and semantic coherence.

4.4 Supervised Fine Tuning

After completing all data processing steps, we proceed with fine-tuning our model. This follows a standard fine-tuning approach, where the loss function is the cross-entropy between the predicted logits and the target summary.

Given that our model contains 0.5 billion parameters and processes extensive text data, computing gradients can be computationally expensive. To mitigate this, we leverage Low-Rank Adaptation (LoRA) [8]. Under this configuration, the updated model weights are defined as:

$$\begin{aligned} Q &= X\left(W_Q + \frac{\alpha}{r} A_Q B_Q\right), \\ K &= X\left(W_K + \frac{\alpha}{r} A_K B_K\right), \\ V &= X\left(W_V + \frac{\alpha}{r} A_V B_V\right), \\ A &= \text{softmax}\left(\frac{1}{\sqrt{d_k}} Q^T K\right) V \end{aligned} \tag{2}$$

where W_Q, W_K, W_V represents the original model weights (frozen), while A_Q, A_K, A_V and B_Q, B_K, B_V are low-rank matrices of rank r. The hyperparameter α is set as $\alpha = 2r$ in our case.

To optimize computational efficiency and memory usage, we perform fine-tuning using the unsloth API. Unsloth outperforms the Hugging Face API in LoRA fine-tuning because it bypasses PyTorch's automatic differentiation. Instead, it manually computes derivatives and integrates them directly into the code for all transformer components, including attention, positional encoding, layer normalization, and MLPs. This optimization reduces computational overhead and significantly speeds up training.

A crucial next step is selecting an appropriate rank r. A higher rank enables greater flexibility in modifying the model's outputs but also increases the risk of overfitting and raises computational costs.

4.4.1 Selecting the Optimal LoRA Rank

Determining the best LoRA rank requires careful tuning. Since hyperparameter optimization for language models is resource-intensive, we limit our search to the rank parameter and perform a grid search.

We train models with adapter ranks $r \in \{8, 16, 32, 64\}$, generating summaries on the validation set. The selection of the optimal r is guided by the evaluation metrics outlined in Section 4.3.

Table 1 presents key evaluation metrics. The column *Number of Words* reports the 25th and 75th percentiles of the distribution of generated word counts. In the boxplot (Figure 3a), these values correspond to the lower and upper edges of each box.

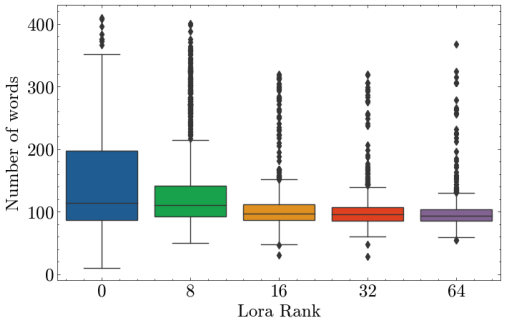
Language On the raw model, approximately 1% of the generated text were not in Porguese language (most of these were in english). Fine tuning solved this problem for all values of r.

Number of words Our prompts asked for a summary of a specific size. This is to avoid the model generating large summaries, which would be costly. As we can see on figure 3a, the raw model has a high variance in the number of generated words, and this is solved for models with lora rank $r > 8$.

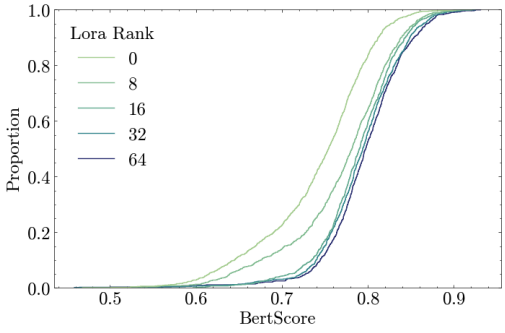
RougeLSum and BertScore We can see that the Rouge metric and BertScore increases monotonically with the Lora Rank, going from 0.29 to 0.41 on Rouge (41% increase) and from 0.77 to 0.80 on BertScore.

LoRA Rank	% of non Portuguese	Number of words	RougeLSum	BertScore
0 (raw model)	1.4 %	[87, 197]	0.29	0.74
8	0.4 %	[92, 141]	0.36	0.77
16	0.3 %	[86, 112]	0.39	0.79
32	0.3 %	[85, 107]	0.40	0.79
64	0.5 %	[85, 103]	0.41	0.80

Table 1. Grid search metrics for each LoRA Rank.



(a) Boxplot of number of generated words in the summary.



(b) Cumulative Density Function of the BertScore.

Figure 3

As our analysis shows, all metrics indicate that the most complex model ($r = 64$) performed best, making it the optimal choice for testing. However, even a significantly lower rank ($r = 8$) already yields substantial improvements across all metrics, highlighting the effectiveness of Supervised Finetuning in adapting the model to specific tasks beyond its original training.

5 Results

5.1 Qualitative analysis

In this section, we conduct a manual sanity check from a human perspective. Using the same example from Section 4.2, we observe that our fine-tuned model produces a significantly improved response compared to the raw model. While the output contains less information than the target summary, it exhibits much better language syntax.

Page Name: Los Angeles Lakers

Finetuned Qwen 0.5B Instruct

The Los Angeles Lakers are an NBA basketball team founded in 1947 in Minneapolis, Minnesota. The team won five league titles before experiencing a decline in attendance. The Lakers have made 56 postseason appearances, secured 33 Pacific Division titles, and won 32 Western Conference championships. The team holds several league records and is considered one of the most popular franchises in the NBA.

5.2 Comparison with other SLMs

We compare the results on the test set with the generated outputs of a `Llama-3.2-1B-Instruct`, with a `Qwen-2.5-3B-Instruct` model and with the raw model `Qwen-2.5-0.5B-Instruct`. Table 2 shows the results accross these models. We can see that comparing to the target summary, the finetuned model is better than the other models, both in Rouge and BertScore metrics.

Model Name	% of non Portuguese	Number of words	RougeLSum	BertScore
Qwen-2.5-0.5B-Instruct	1.4 %	[87, 214]	0.29	0.74
Llama-3.2-1B-Instruct	0.1 %	[65, 86]	0.37	0.78
Qwen-2.5-3B-Instruct	**0.0** %	[75, 87]	0.35	0.79
Finetuned Model	0.4 %	[85, 103]	**0.41**	**0.80**

Table 2. Generation metrics for each model.

Indeed, our model successfully replicates the patterns of the target summary, as it was specifically trained for that purpose. However, it is crucial to determine whether the summaries are genuinely better, rather than just more aligned with the target. To assess this, we utilize a larger model, `Llama-3-8B-Instruct`, to compare summaries generated by our Finetuned model against those from other models, capturing the winning rates.

Table 3 demonstrates that our model consistently outperforms the baseline `Qwen-2.5-0.5B-Instruct`, achieving a 61% winning rate. Additionally, its performance is comparable to a model twice its size (`Llama-3.2-1B-Instruct`), with a 51% winning rate. However, when compared to a larger 3B model (`Qwen-2.5-3B-Instruct`), our model falls behind, achieving only a 22% winning rate.

Model Name	Winning rate
Qwen-2.5-0.5B-Instruct	61 %
Llama-3.2-1B-Instruct	51 %
Qwen-2.5-3B-Instruct	22 %

Table 3. Winning rates of the Finetuned model against different models

6 Conclusion

Language Models are the state of the art in almost all NLP tasks. These models are trained on a very large corpora of data and can solve very complex tasks automatically. However, due to its complex operations and dimension, these models can generate unexpected outputs. We observe that smaller versions of these models tends to be more prone to this behaviors, since it has less parameters and less "knowledge".

In this work we performed fine tuning in a small language models in summarization task in Portuguese on the `Qwen2.5-0.5B-Instruct` model.

We begin by carefully collecting data in Portuguese and generating target summaries using a larger model. However, we observe that this model struggles with the task, often producing text in multiple languages and containing repetitive ideas. To improve performance, we fine-tune the model using LoRA, adjusting the LoRA Rank. Our results indicate that the model successfully learns the task. Finally, we compare the fine-tuned model to its untrained version and larger models. Through an LLM-as-a-judge experiment, we confirm that the trained model performs comparably to a model twice its size but remains behind a 3B-parameter model.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[2] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[3] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif: Scaling reinforcement learning from human feedback with ai feedback, 2023.

[4] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024.

[5] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

[6] Kavita Ganesan. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks, 2018.

[7] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

A Computational ressources

We conducted all experiments using a single Nvidia RTX A5000 GPU, which is part of the school's computational resources. This GPU has approximately 24GB of memory. To comply with the 15GB memory constraint set by the project, we limited our code to 60% of the total GPU memory using the following command:

```
torch.cuda.set_per_process_memory_fraction(0.6, device=0)
```

B Prompts used in the experiments

```
 You are a virtual assistant tasked with generating summaries of texts
in Portuguese.  Your summary should be no longer than {n_words} words and
must include all the main information from the text.

Here is the text:

{text}

Provide a summary of the text above in no more than {n_words} words.
```

C Hyperparameters

In our experiments, we fine-tuned the language model using the following hyperparameters:

- **Learning Rate:** 2e-5
- **Optimizer:** AdamW
- **Weight Decay:** 0.01
- **Number of Training Epochs:** 4
- **Maximum Sequence Length:** 6000 tokens
- **Warm-up Steps:** 5
- **Learning Rate Scheduler:** Linear decay
- **Dropout Rate:** 0.05
- **Decoding strategy:** Greedy Decoding
- **Maximum of generated tokens:** 500

With this hyperparameters, figure 4 shows the evolution of the Loss function on the train set.

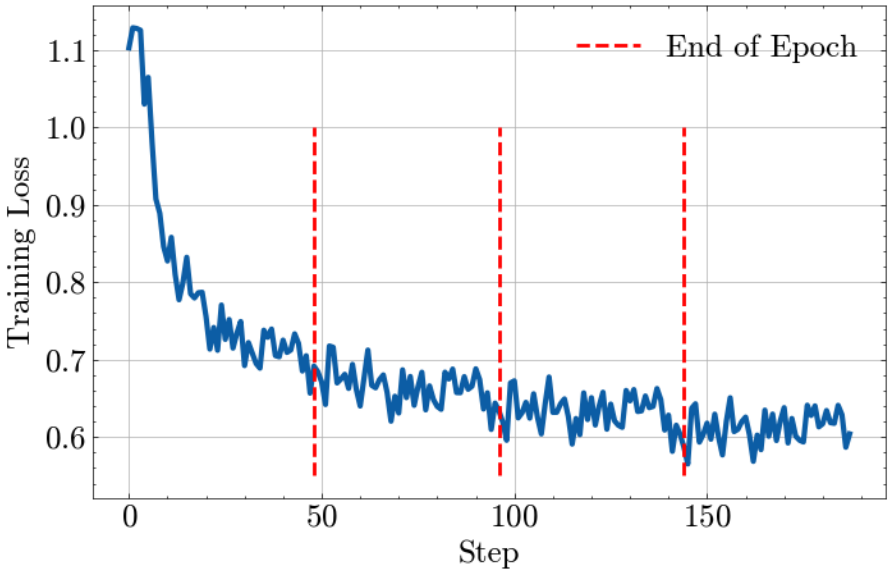


Figure 4. Loss on the train set over training steps.