The Joy and Pain of Training an LLM from Scratch

A Technical Report on the Development of the Zagreus and Nesso Model Families

Anonymous Author[1]

[1]Institution, Country

Abstract

Training a fully functional modern neural network, specifically a Large Language Model (LLM), from first principles has been a foundational ambition since the inception of our community *mii-llm* [1] (Made in Italy – Large Language Model). This technical report documents the development of the Zagreus and Nesso model families, spanning motivation, framework selection, data engineering, pre-training, post-training, and a comprehensive set of reproducible evaluations across multiple languages.

Keywords

Small Language Models, Italian NLP, Pre-training, Post-training, Evaluation, Reproducibility

1. Table of Contents

2. 1. Motivation: The Vision of Sovereign Edge Intelligence

Training a fully functional modern neural network, specifically a Large Language Model (LLM), from first principles has been a foundational ambition since the inception of our community mii-llm [1] that stands for Made in Italy – Large Language Model.

In the current landscape, the convergence of distributed computing power and accessible knowledge has never been more potent; consequently, constructing an intelligent machine stands as one of the most exciting tasks a group of machine learning specialists can undertake. This vision materialized when Antonio Baldassarra (CEO of Seeweb) and Marco Cristofanilli (Head of AI at Seeweb) commissioned us to develop a Small Language Model (SLM) from scratch utilizing the Seeweb infrastructure [2].

Seeweb, a cloud provider with a strategic focus on AI, granted us access to a cluster of on-demand nodes comprising a total of 64 NVIDIA A100 GPUs. Our primary objective was to experiment and deliver a state-of-the-art SLM with approximately 500 million parameters, built from the ground up and optimized for edge use cases within the Italian language ecosystem.

We hypothesize that, in the coming years, intelligent devices—and virtually any hardware equipped with a chip—will be enhanced by neural architectures with embedded reasoning and language capabilities. Small, efficient models will be key to enabling automation at the edge. To address this need, we created Zagreus, arguably one of the few high-performing small language models dedicated to the Italian language.

In the spirit of open and reproducible research, we are releasing the full Zagreus and Nesso lineup: seven models in total—four base (pretrained) checkpoints for bilingual models and three post-trained variants. Notably, our post-trained models are designed to compete on standard benchmarks with state-of-the-art models of comparable size, demonstrating that carefully engineered small models can achieve near frontier-level performance within their parameter class.

Base models:

- `zagreus-0.4B-base-ita`: English Italian bilingual model
- `zagreus-0.4B-base-spa`: English Spanish bilingual model
- `zagreus-0.4B-base-por`: English Portuguese bilingual model
- `zagreus-0.4B-base-fra`: English French bilingual model

Post-trained models:

- `Nesso-0.4B-instruct`: English Italian for conversational use cases
- `Nesso-0.4B-agentic`: English Italian for agentic and function calling use cases
- `Open-Zagreus-0.4B`: Fully open source data used to train this model

We are releasing this detailed blog post, covering every step and data point required to reproduce the project, as we strongly believe in the importance of open source in reducing technological and geopolitical dependencies.

3. 2. Technology Stack: Framework Selection

There are numerous frameworks available for creating an LLM from scratch. We conducted a comparative analysis of several options. Below is a summary of our testing and the rationale behind our ultimate decision to utilize Nanotron by Hugging Face [3].

3.1. Framework Comparative Analysis

Megatron-LM: Developed by NVIDIA, this is a powerful framework designed for training large transformer models with billions of parameters [4]. While it is likely an optimal choice for large, well-resourced teams, we found it challenging to set up and deploy effectively on our specific cluster infrastructure.

Llama-Factory: A versatile and user-friendly open-source framework that simplifies fine-tuning, training, and deployment of a wide range of LLMs [5]. However, our evaluation suggests it is more specialized for fine-tuning than for pre-training from scratch.

nanoGPT and nanochat: Both created by Andrej Karpathy, these projects prioritize simplicity and educational value. nanoGPT [6] is a minimalist, readable codebase designed as a learning tool, though it is now considered deprecated in favor of its successor. nanochat [7] is the evolution of nanoGPT, offering a full-stack, end-to-end pipeline for building a complete ChatGPT-like chatbot. It covers the entire lifecycle, from tokenization and pre-training to fine-tuning and a web interface, all within a compact and hackable codebase.

Although nanochat had not yet been released when we commenced this project, we believe it has a promising future, especially given its recent integration into the Transformers library.

3.2. Our Choice: Hugging Face Nanotron

Ultimately, we selected Hugging Face Nanotron [3]. It is a minimalistic library focused on 3D parallelism (Data, Tensor, and Pipeline) specifically for pre-training transformer models. We value Hugging Face for its commitment to openness.

We found the library well-suited for multi-node training; furthermore, it is natively integrated into the Hugging Face ecosystem (Accelerate, Datasets, hf-cli), ensuring that workflows—from data tokenization to model release—remain cohesive. During the development cycle, we identified minor bugs and are actively contributing to the library via Pull Requests. We also established a fork of Nanotron optimized to run directly on a Slurm cluster [8].

4. 3. Data Engineering: The Tokenization Pipeline

Data is the *sine qua non* for creating an LLM. The volume of data required is contingent upon the target model size and the available compute budget. Operating as a GPU-constrained team—and thanks to the sponsorship from Seeweb—we chose to build a small language model of ∼500 million parameters, trained on approximately 1 trillion tokens.

4.1. Dataset Sources

We utilized exclusively open source datasets by the Hugging Face team for creating our four bilingual foundational model released.

Below is the data distribution per model:

mii-llm/nesso-0.4B-ita:

- https://huggingface.co/datasets/HuggingFaceFW/fineweb/viewer/sample-350BT (350 billion tokens) [9]
- https://huggingface.co/datasets/HuggingFaceFW/fineweb-2/viewer/ita_Latn [10]
- https://huggingface.co/datasets/HuggingFaceFW/finepdfs/viewer/ita_Latn [11]
- https://huggingface.co/datasets/bigcode/starcoderdata (250 billion tokens) [12]

mii-llm/nesso-0.4B-fra:

- https://huggingface.co/datasets/HuggingFaceFW/fineweb/viewer/sample-350BT (350 billion tokens) [9]
- https://huggingface.co/datasets/HuggingFaceFW/fineweb-2/viewer/fra_Latn [10]
- https://huggingface.co/datasets/HuggingFaceFW/finepdfs/viewer/fra_Latn [11]
- https://huggingface.co/datasets/bigcode/starcoderdata (250 billion tokens) [12]

mii-llm/nesso-0.4B-por:

- https://huggingface.co/datasets/HuggingFaceFW/fineweb/viewer/sample-350BT (350 billion tokens) [9]
- https://huggingface.co/datasets/HuggingFaceFW/fineweb-2/viewer/por_Latn [10]
- https://huggingface.co/datasets/HuggingFaceFW/finepdfs/viewer/por_Latn [11]
- https://huggingface.co/datasets/bigcode/starcoderdata (250 billion tokens) [12]

mii-llm/nesso-0.4B-spa:

- https://huggingface.co/datasets/HuggingFaceFW/fineweb/viewer/sample-350BT (350 billion tokens) [9]
- https://huggingface.co/datasets/HuggingFaceFW/fineweb-2/viewer/spa_Latn [10]
- https://huggingface.co/datasets/HuggingFaceFW/finepdfs/viewer/spa_Latn [11]
- https://huggingface.co/datasets/bigcode/starcoderdata (250 billion tokens) [12]

4.2. The Tokenization Process

Raw datasets are not ready for immediate training; they must first be tokenized. Tokenization is a CPU intensive process that transforms text strings into token sequences (numerical IDs). As a rule of thumb for storage estimation, for every 1 GB of text, approximately 3 GB of tokenized outputs are generated. For $\sim$1 trillion tokens, one typically requires at least 3 to 5 terabytes of disk space (depending on format, sharding strategy, and compression).

We selected the Llama-3.2 tokenizer (from the Llama-3.2-1B model) because its multilingual tokenization capabilities are robust and widely adopted. Using the datatrove library [13], the process took over three weeks of continuous computation to generate $\sim$1 trillion tokens, stratified as roughly 400B English, 400B Italian, and 200B Code.

5. 4. Pre-training: The Core Engine

Pre-training is the foundational step in building an LLM, transforming raw tokenized data into a model capable of context aware text completion. This is the most time consuming and GPU intensive phase. While massive models may require thousands of GPUs, our sub 1 billion parameter model was effectively trained on the 64 GPU cluster provided by Seeweb.

We utilized Nanotron [3], which supports multiple architectures, including Llama-3.2, Qwen-2.5, and Mixture-of-Experts (MoE) variants. For this project, we adopted a modified Llama-3.2 fully dense architecture. Our design choice was motivated by the hypothesis that, in the small-parameter regime ($\sim$500M parameters), fully dense models provide better compute utilization and more stable training dynamics than sparse architectures such as MoE.

In tightly constrained capacity settings, the routing overhead and expert under-utilization typical of MoE architectures may offset their theoretical efficiency advantages. Working with a GPU cluster is streamlined by HPC tools; we employed the Slurm scheduler. Slurm allows the cluster to be viewed as a unified Linux system where jobs can be executed across many GPUs in parallel, while handling checkpoints and logs in real time.

The most challenging aspect remains ensuring the software stack—from drivers and CUDA/NCCL to Python libraries—functions harmoniously, often requiring resolution of version and ABI incompatibilities. Successfully running a distributed training job on the tokenized data was a profound milestone. Observing the loss curve decrease from raw data after days of waiting conveys the sense of operating at the edge of scientific and engineering capability—a genuinely intense moment for a researcher.

For out-of-the-box functionality, we recommend our fork: https://github.com/mii-llm/nanotron [8] (a fork of https://github.com/huggingface/nanotron/ [3]), pending the merge of our Pull Request.

6. 5. Post-Training: Shaping Behavior

Creating a base model from scratch represents a major technical achievement, and we consider this work a contribution to the open community. However, a foundation model alone—even with a fully reproducible pipeline and transparent data distribution—is rarely sufficient for direct real-world deployment. The post-training phase is responsible for shaping the model's behavior toward practical usability.

This phase typically requires significantly fewer GPUs and a smaller data volume compared to pre-training. However, the *quality* and *curation strategy* of the data become substantially more important than raw scale.

Over the past several years, we have post-trained models for domain-specific applications including finance, cybersecurity, structured function calling, and agentic execution patterns. Through this work, we have curated a substantial internal dataset collection that enables controlled experimentation across varied instruction-following regimes. This dataset collection, built with meticulous care and long-term iteration, constitutes a strategic asset for our research group.

For this reason, we have decided not to publish it as open source, as we consider it a competitive advantage. Nevertheless, we believe that releasing the trained models and all evaluation results provides significant value to the broader community. Most importantly, we demonstrate that we have been able to build and release a model that performs competitively head to head with state of the art models of similar parameter scale.

We are releasing three primary post-trained models:

- **Nesso-0.4B-instruct**: optimized for conversational and instruction-following use cases.
- **Nesso-0.4B-agentic**: optimized for function calling, structured outputs, and agentic execution patterns.

Both models utilize **Nesso-0.4B-ita** as the base and are trained on a bilingual corpus (English/Italian). It is important to note that both models are currently at the **SFT (Supervised Fine-Tuning)** stage.

In the coming weeks, we will execute the **DPO (Direct Preference Optimization)** stage and subsequently update both the models and their evaluation results. We also released a third, fully open model: **Open-Zagreus-0.4B**.

Thanks to the work of the Italian open-source community **mii-llm** [1], and in particular Michele Montebovi who published the SFT dataset *OpenItalianData*—all data used and all training recipes for this model are fully open and reproducible as a full open source model from data to weights.

7. 6. Pre-trained Foundational Models Evaluations

This section presents quantitative evaluations of our pre-trained foundational models. We include multiple data points to demonstrate how our data curation strategy and architectural configuration enabled the training of competitive small language model families. These results serve both as validation and as a reproducible baseline for future experiments.

We are contributors to **lm-evaluation-harness** [14] for multilingual benchmarks and relied extensively on this framework. For each benchmark, we provide the exact command used to ensure the evaluation reproducibility.

7.1. Zagreus-0.4B-ita-base

Evaluation Command

```
lm-eval --model hf --model_args pretrained=checkpoint \
  --tasks m_mmlu_it --num_fewshot 5 --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks hellaswag_it,arc_it --device cuda:0 --batch_size 1
```

Table 1
Zagreus-0.4B-ita-base checkpoint progression.

Checkpoint	mmlu_it (acc)	hellaswag_it (acc_norm)	arc_it (acc_norm)	Media
v2-95k	0.2529	0.3366	0.2652	0.2849
v2-205k	0.2628	—	—	0.2628
v2-290k	0.2428	0.3492	0.2335	0.2752
v2-305k	0.2598	0.3562	0.2652	0.2937
v2-365k	0.2566	0.3664	0.2712	0.2981
v2-390k	0.2556	0.3438	0.2498	0.2831
v2-460k	0.2540	0.3778	0.2549	0.2956
v2-520k	0.2540	0.3778	0.2549	0.2956
v2-590k	0.2547	0.3651	0.2455	0.2884
v2-630k	0.2562	0.3632	0.2643	0.2946
v2-680k	0.2538	0.3740	0.2592	0.2957
v2-775k	0.2535	0.3750	0.2583	0.2956

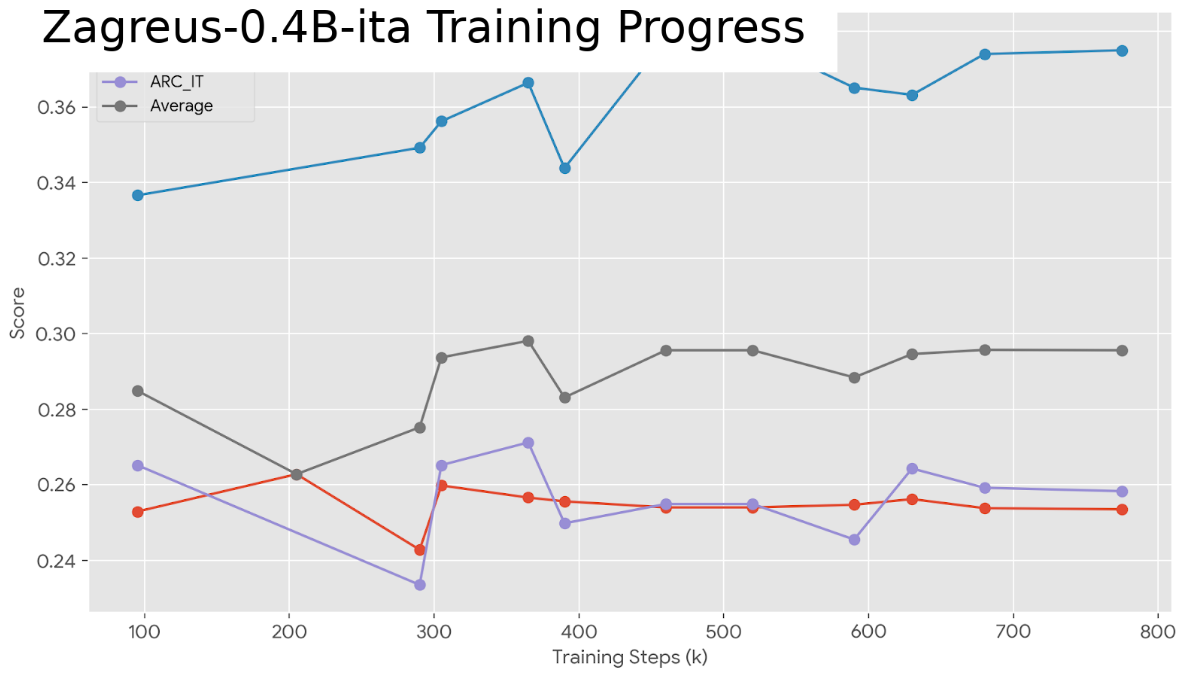

Figure 1: zagreus-ita.

Checkpoint progression

7.2. Zagreus-0.4B-spa-base (Spanish)

Evaluation Command

```
lm-eval --model hf --model_args pretrained=checkpoint \
  --tasks m_mmlu_es --num_fewshot 5 --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks hellaswag_es,arc_es --device cuda:0 --batch_size 1
```

Table 2
Zagreus-0.4B-spa-base results across training steps.

Steps	mmlu_es	arc_es	hellaswag_es	Average
146k	0.254	0.265	0.409	0.309
216k	0.237	0.270	0.414	0.307
292k	0.254	0.262	0.417	0.311
406k	0.254	0.269	0.423	0.315
518k	0.255	0.280	0.429	0.321

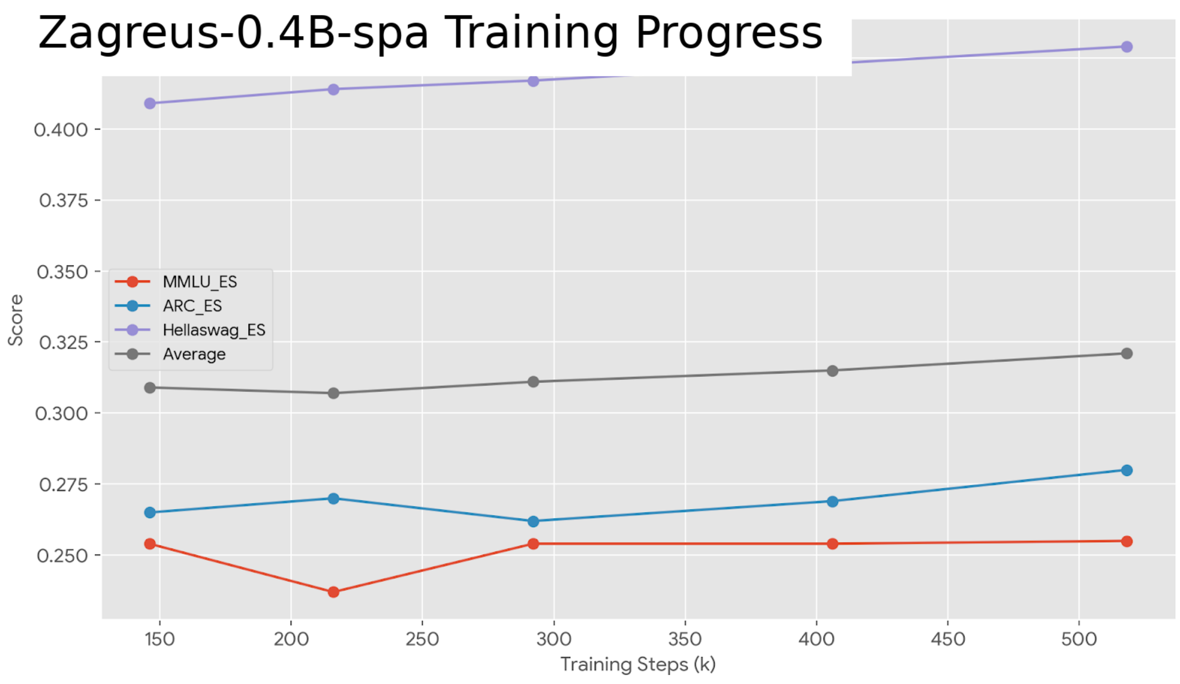

Figure 2: zagreus-spa.

Table 3
Zagreus-0.4B-fra results across training steps.

Steps	m_mmlu_fr	arc_fr	hellaswag_fr	Average
129k	0.262	—	—	0.262
231k	0.263	—	—	0.263
365k	0.256	0.278	0.414	0.316
456k	0.267	—	—	0.267
603k	0.256	0.278	0.414	0.316
705k	0.266	0.281	0.417	0.321

7.3. Zagreus-0.4B-fra (French)

Evaluation Command

```
lm-eval --model hf --model_args pretrained=checkpoint \
  --tasks m_mmlu_fr --num_fewshot 5 --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks hellaswag_fr,arc_fr --device cuda:0 --batch_size 1
```

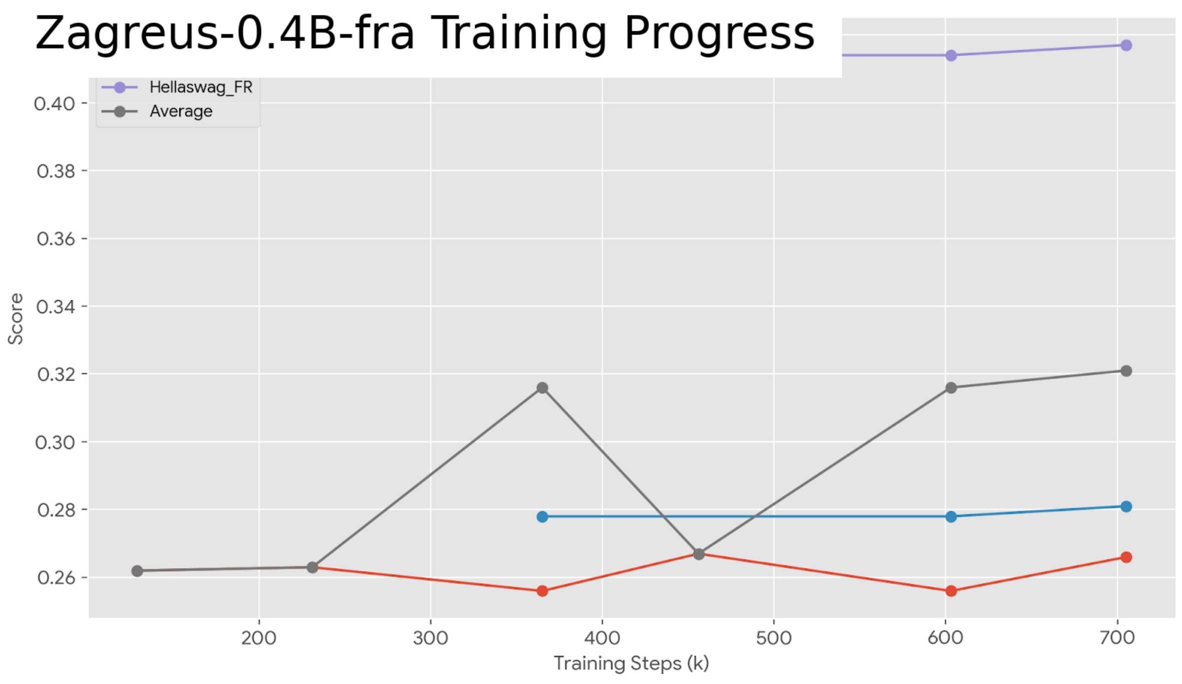

Figure 3: zagreus-fra.

Table 4
Zagreus-0.4B-por results across checkpoints.

Checkpoint	ARC	HellaSwag	MMLU	Media
153k	0.2667	0.3732	0.2685	0.3028
207k	0.2705	0.3768	0.2671	0.3048
276k	0.2718	0.3789	0.2664	0.3057
345k	0.2564	0.3796	0.2669	0.3009
414k	0.2682	0.3842	0.2673	0.3066
483k	0.2667	0.3865	0.2658	0.3063
582k	0.2786	0.3865	0.2688	0.3113

Evaluation procedure identical to previous sections.

7.4. Zagreus-0.4B-por (Portuguese)

Evaluation Command

```
lm-eval --model hf --model_args pretrained=checkpoint \
  --tasks m_mmlu_pt --num_fewshot 5 --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks hellaswag_pt,arc_pt --device cuda:0 --batch_size 1
```

7.5. lm-evaluation-harness-pt

For portuguese base model we also evaluate against the fantastic work of Eduardo Garcia and a fork of lm-eval [15] that has also an important leaderboard [16] comparing many open source models. Below the results and the comparison with Qwen3-0.6B-Base.

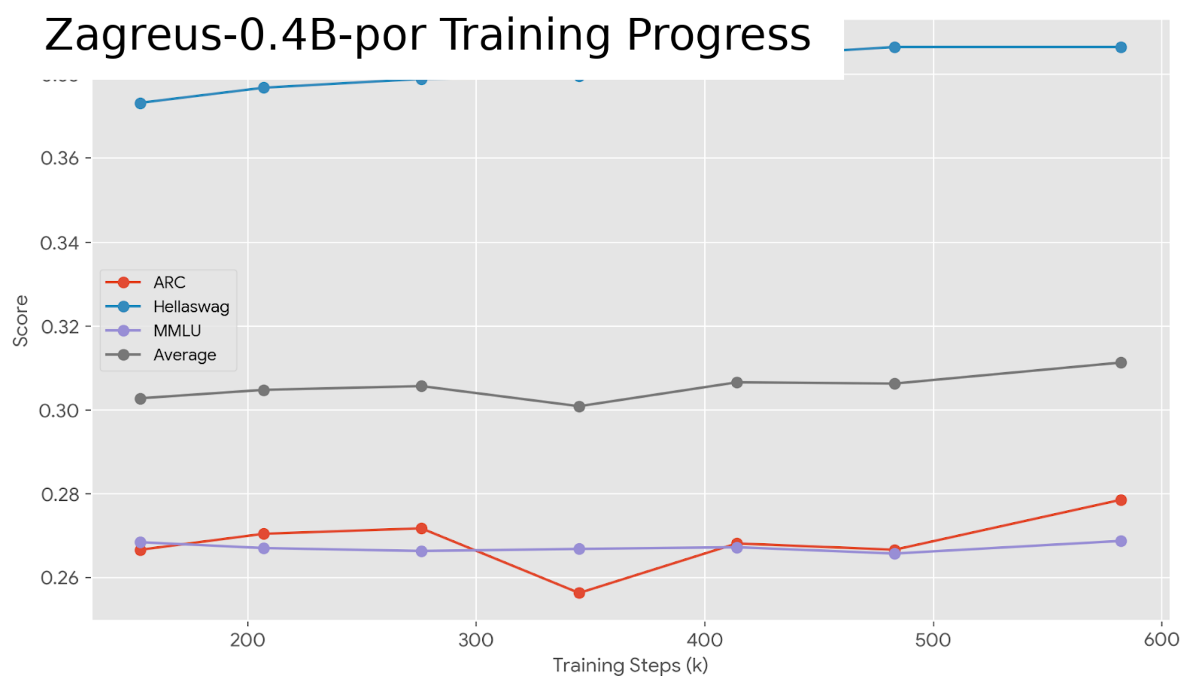

Figure 4: zagreus-por.

Table 5
Portuguese leaderboard-style evaluation and comparison (lm-evaluation-harness-pt).

Model / Checkpoint	RTE	STS	BLUEX	ENEM	FAQUAD NLI	HateBR	OAB	PT Hate	TweetSent	Media
zagreus 483k	0.4624	0.1650	0.2434	0.2071	0.4397	0.3327	0.2528	0.4817	0.3220	0.3230
zagreus 582k	0.3361	0.0449	0.2100	0.1903	0.4397	0.3825	0.2392	0.4444	0.1542	0.2713
Qwen3-0.6B-Base	0.3333	0.0726	0.1057	0.0077	0.4397	0.3333	0.0428	0.4123	0.5646	0.2569

Evaluation Command

```
lm_eval --model huggingface \
  --model_args "pretrained=giux78/zagreus-3B-165000,revision=main" \
  --tasks enem_challenge,bluex,oab_exams,assin2_rte,assin2_sts,faquad_nli,
      hatebr_offensive,portuguese_hate_speech,tweetsentbr \
  --device cuda:0 \
  --output_path "./"
```

8. 7. Post-Trained Nesso Models Evaluations

Evaluation Commands

```
lm-eval --model hf --model_args pretrained=checkpoint \
  --tasks m_mmlu_it --num_fewshot 5 --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=checkpoint \
  --tasks mmlu --num_fewshot 5 --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks hellaswag_it,arc_it --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
```

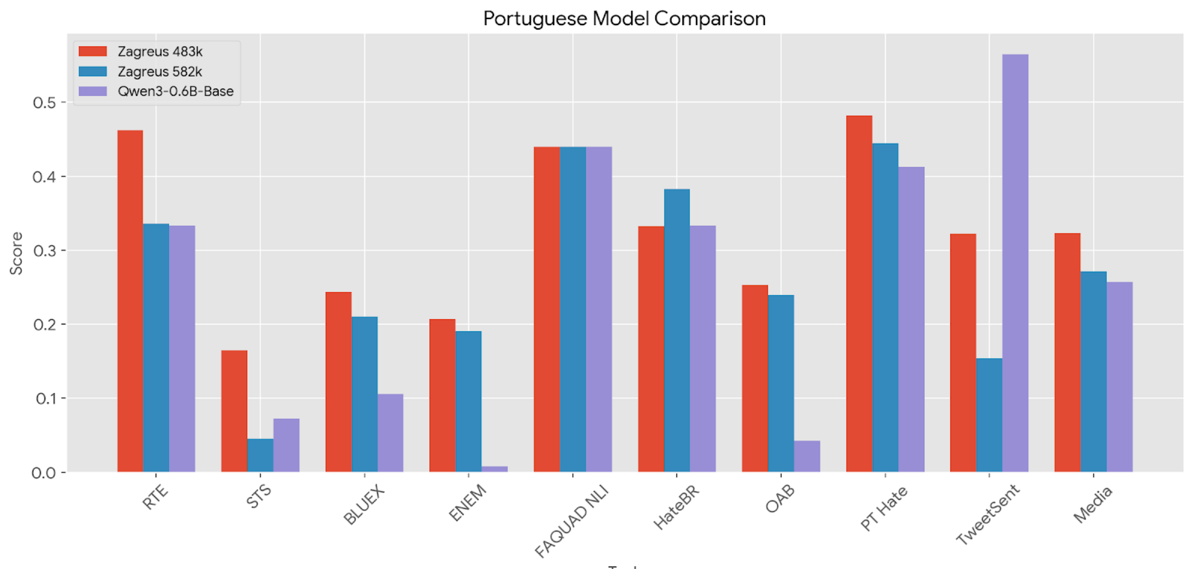

Figure 5: por-garcia.

Table 6
English/Italian benchmark comparison across post-trained and baseline models.

Model	IFEval EN	ARC_EN	HS_EN	MMLU_EN	Media EN	IFEval IT	ARC_IT	HS_IT	MMLU_IT	Media IT	Media Totale
Qwen/Qwen3-0.6B	0.2758	0.3430	0.4742	0.4013	0.3736	0.3058	0.2729	0.3598	0.4025	0.3353	0.3545
Nesso-04B-instruct	0.3465	0.3003	0.4629	0.2871	0.3492	0.2962	0.2874	0.4076	0.2875	0.3197	0.3345
Nesso-04B-agentic	0.2962	0.2534	0.4062	0.2889	0.3112	0.2914	0.2541	0.3673	0.2730	0.2965	0.3039
LiquidAI/LFM2-350M	0.1595	0.2457	0.3092	0.3445	0.2647	0.1427	0.2464	0.2994	0.3132	0.2504	0.2576

```
  --tasks hellaswag,arc --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks ifeval-ita --device cuda:0 --batch_size 1

lm-eval --model hf --model_args pretrained=LiquidAI/LFM2-350M \
  --tasks ifeval --device cuda:0 --batch_size 1
```

Discussion

As observed, Qwen maintains a clear advantage on MMLU (both English and Italian). However, across several other benchmarks—particularly instruction-following and reasoning-oriented tasks—Nesso achieves competitive or superior performance. Considering that MMLU is a widely used and often saturated benchmark, frequently incorporated into training corpora, we believe our results demonstrate that we have created a highly competitive small language model optimized for English/Italian edge inference scenarios.

9. Open-Nesso-0.4B Evaluation

Open-Nesso-0.4B-ita is our fully open-source variant. It is based on Nesso-0.4B-ita and trained on the publicly available dataset published by Michele Montebovi.

Download: https://huggingface.co/datasets/DeepMount00/OpenItalianData [17]

The model and dataset demonstrate that it is possible to build competitive English Italian language models using exclusively open-source resources.

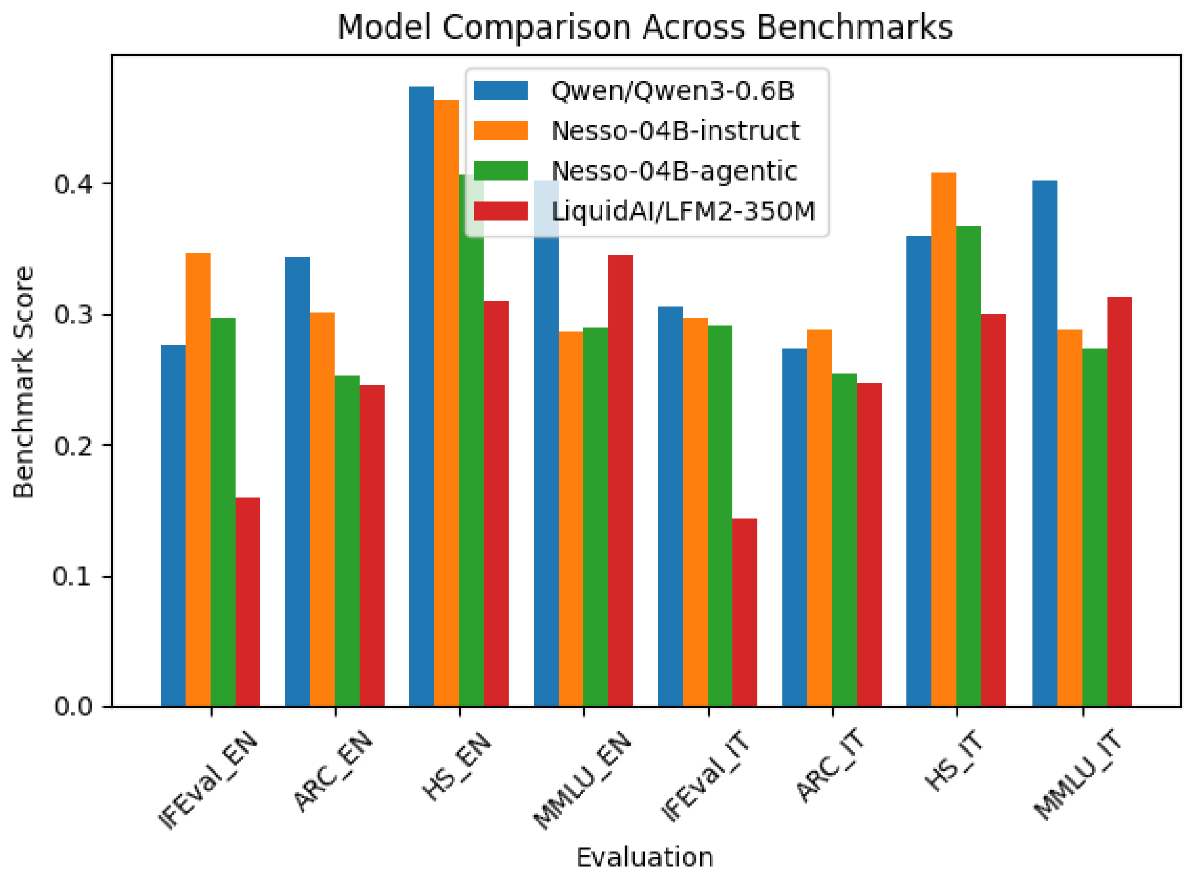

Figure 6: nesso-all.png

Table 7
Open-Nesso-0.4B evaluation (Italian).

Model	mmlu_it	arc_it	hellaswag_it	Media
giux78/open-zagreus-350M-sft	0.2530	0.3020	0.3608	0.3053

10. 8. Conclusion

The *Zagreus and Nesso Model Families* project stands as a remarkable and highly important contribution to the field of language model research, particularly within the realm of **Small Language Models (SLMs)**. At a time when the community is largely focused on scaling models ever larger, this work demonstrates that **starting from scratch and engineering a small, efficient model can be both feasible and impactful**.

The initiative directly addresses the critical need for models that are capable of **intelligent reasoning at the edge**, optimally suited for deployment on everyday devices with limited compute resources — a paradigm that will only grow in strategic importance as AI becomes more ubiquitous across hardware platforms.

This report does not merely describe a model; it **documents the entire empirical journey** of developing SLMs from first principles, from motivation and data engineering to pre-training, validation, and deployment. By releasing **seven distinct models**, including multilingual foundational checkpoints and post-trained variants optimized for conversational and agentic use cases, the project sets a new standard for reproducibility and openness in LLM research.

Importantly, the work emphasizes that **carefully engineered small models can match or ap-**

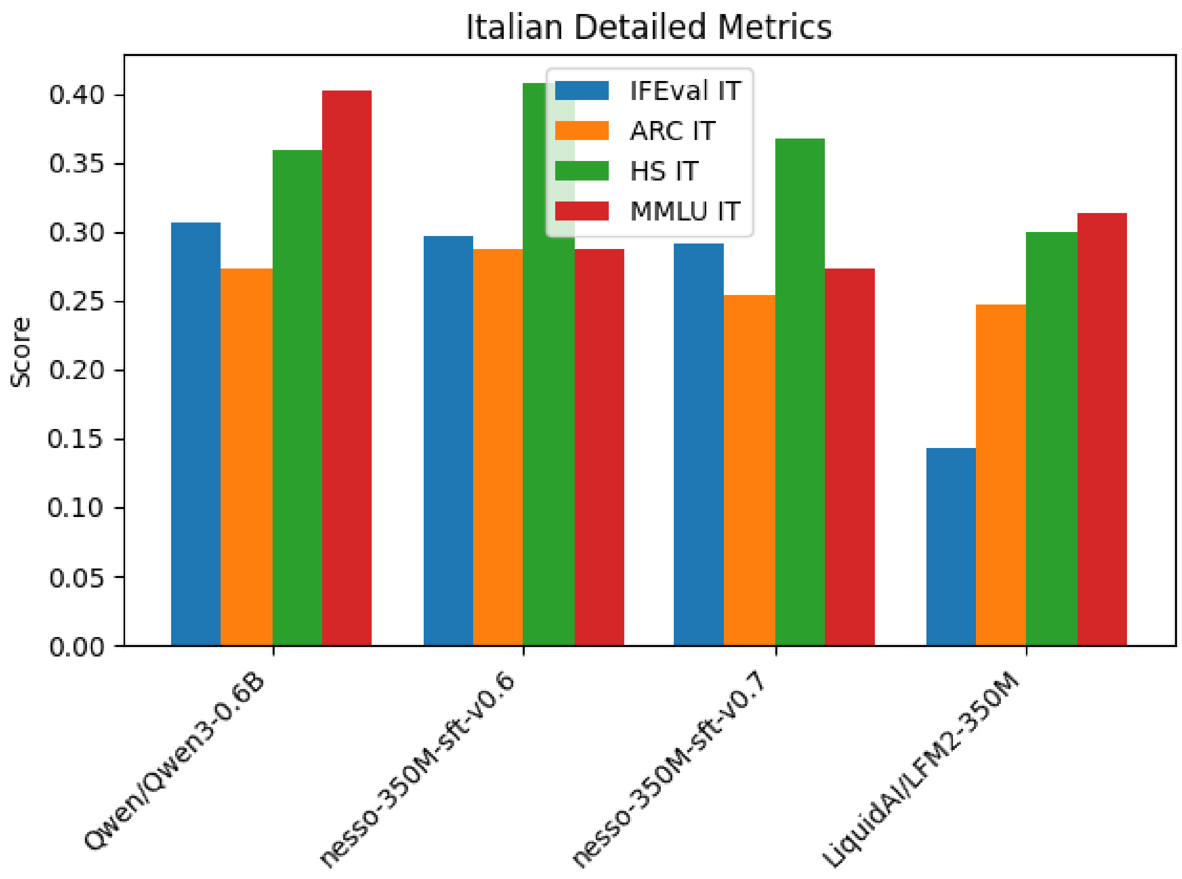

Figure 7: italian-nesso.png

proach the performance of much larger counterparts on standardized benchmarks, underlining a strategic shift in how the community can think about computational efficiency without sacrificing capability.

In addition to the scientific and technical achievements, the *Zagreus–Nesso SLM* effort embodies a broader philosophical commitment: the advancement and **democratization of AI through open research and open source tools**. By providing detailed data pipelines, architectural choices, and a transparent account of trade-offs encountered in training at scale, this work becomes an invaluable resource for anyone seeking to replicate or build upon it.

Therefore, the importance of this report lies not only in its immediate results but also in its **lasting influence on how future small language models may be conceived, trained, and deployed**.

Acknowledgments

We thank the open-source community and all contributors who made evaluation tooling and datasets broadly accessible, enabling reproducible benchmarking.

Declaration on Generative AI

During the preparation of this work, the author(s) used generative AI tools for grammar and spelling refinement. After using these tool(s), the author(s) reviewed and edited the content as needed and take full responsibility for the publication's content.

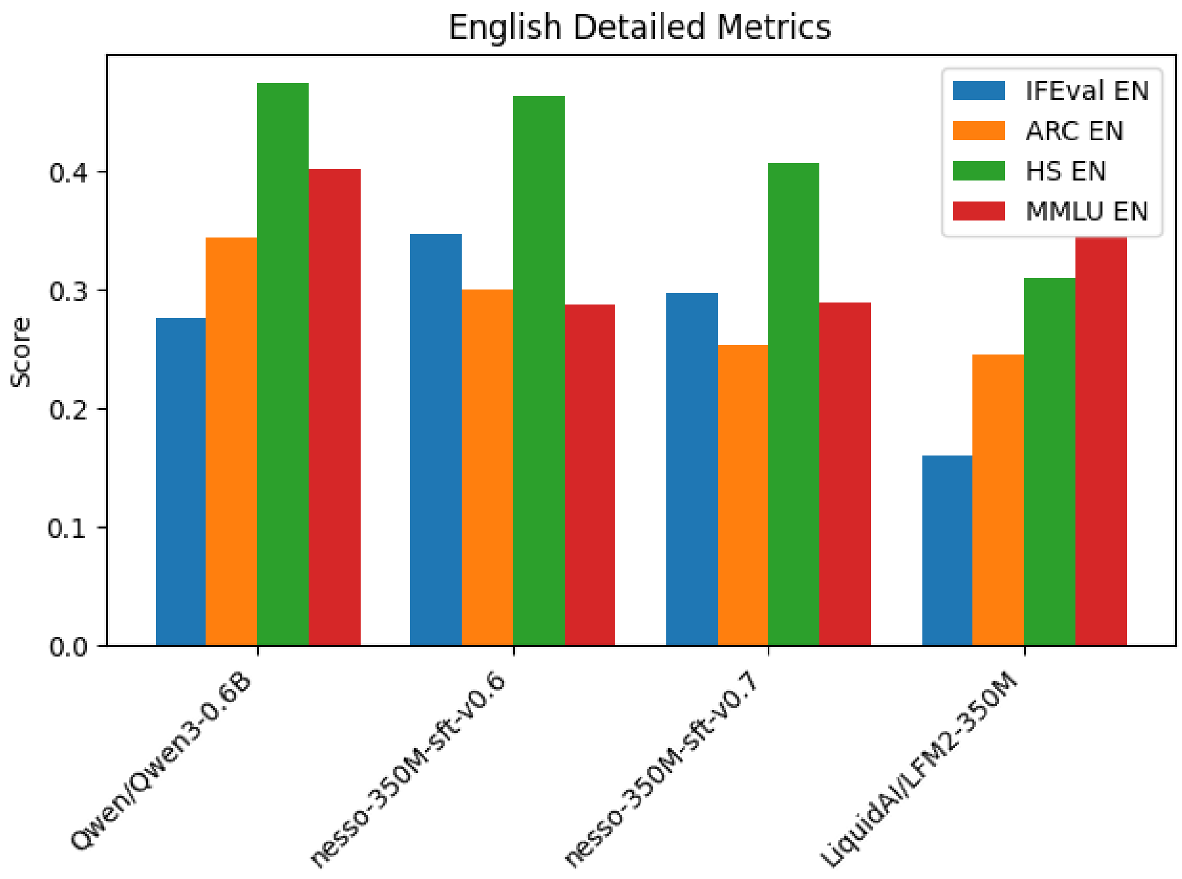

Figure 8: english$_nesso.png$

References

[1] mii-llm (made in italy – large language model), https://mii-llm.ai, ???? Accessed 2026-02-17.
[2] Seeweb, https://www.seeweb.it, ???? Accessed 2026-02-17.
[3] Hugging face nanotron, https://github.com/huggingface/nanotron, ???? Accessed 2026-02-17.
[4] Megatron-lm, https://github.com/NVIDIA/Megatron-LM, ???? Accessed 2026-02-17.
[5] Llama-factory, https://github.com/hiyouga/LLaMA-Factory, ???? Accessed 2026-02-17.
[6] nanogpt, https://github.com/karpathy/nanoGPT, ????. Accessed 2026-02-17.
[7] nanochat, https://github.com/karpathy/nanochat, ????. Accessed 2026-02-17.
[8] mii-llm nanotron fork, https://github.com/mii-llm/nanotron, ????. Accessed 2026-02-17.
[9] Huggingfacefw/fineweb, https://huggingface.co/datasets/HuggingFaceFW/fineweb/viewer/sample-350BT, ????. Accessed 2026-02-17.
[10] Huggingfacefw/fineweb-2, https://huggingface.co/datasets/HuggingFaceFW/fineweb-2, ????. Accessed 2026-02-17.
[11] Huggingfacefw/finepdfs, https://huggingface.co/datasets/HuggingFaceFW/finepdfs, ????. Accessed 2026-02-17.
[12] bigcode/starcoderdata, https://huggingface.co/datasets/bigcode/starcoderdata, ???? Accessed 2026-02-17.
[13] datatrove, https://github.com/huggingface/datatrove, ???? Accessed 2026-02-17.
[14] Eleutherai lm-evaluation-harness, https://github.com/EleutherAI/lm-evaluation-harness, ???? Accessed 2026-02-17.
[15] eduagarcia/lm-evaluation-harness-pt, https://github.com/eduagarcia/lm-evaluation-harness-pt, ???? Accessed 2026-02-17.

[16] Open portuguese llm leaderboard, https://huggingface.co/spaces/eduagarcia/open_pt_llm_leaderboard, ????. Accessed 2026-02-17.

[17] Deepmount00/openitaliandata, https://huggingface.co/datasets/DeepMount00/OpenItalianData, ????. Accessed 2026-02-17.