DEVELOPMENT AND IMPLEMENTATION OF A SNOMED CT-SPECIALIZED LARGE LANGUAGE MODEL VIA QLORA FINE-TUNING AND LOCAL DEPLOYMENT WITH OLLAMA

[1]DANIEL ANTOKOLETZ HUERTA, [2]FERNANDO MARIO LA ROSA

[1]Universidad de Moron School of Engineering, Computer Science, and Agri-Food Sciences Moron
[2]HL7 Argentina - Buenos Aires, Argentina Daniel Antokoletz Huerta
E-mail: dantokoletzhuerta@unimoron.edu.ar, flarosa@unimoron.edu.ar
https://orcid.org/0009-0007-6351-9431, https://orcid.org/0009-0001-0506-2257

Abstract - This article describes the complete development of a large language model (LLM) specialized in the clinical terminology of the SNOMED CT standard, covering three phases: training corpus construction, fine-tuning using Quantized Low-Rank Adaptation (QLoRA), and local deployment with Ollama. The final corpus (v5_final) comprises 2,186,183 examples across five semantic types generated from the SNOMED CT Release Format 2 (RF2) files and presents a statistically uniform distribution with a P95/mean ratio of 1.78 tokens — comparable to standard NLP corpora. The complete fine-tuning on this corpus is currently in progress, with approximately 60 days of processing remaining. Initial tests with a reduced corpus of 50,000 examples, carried out on Mistral-7B-Instruct-v0.2 using consumer-grade hardware (NVIDIA Turing GPU CC 7.5, 6 GB VRAM, float32), achieved a minimum loss of 0.370 and a token accuracy of 91.5%. We document the technical implementation errors, the numerical instability observed during checkpoint resumptions when processing the full corpus, and the formulation of the Cold-Start Optimizer Resumption (CSOR) framework [1] as a solution to the incompatibility between the AdamW optimizer state and the restarted learning rate scheduler. The specialized model is integrated into the Ollama ecosystem for local inference without reliance on cloud services.

Keywords - SNOMED CT, QLoRA, LoRA, Mistral-7B, fine-tuning, Ollama, CSOR, AdamW, clinical NLP, medical terminology, consumer GPU, fp32, local deployment.

I. INTRODUCTION

Standardized clinical terminology is a critical component in health information systems. SNOMED CT (Systematized Nomenclature of Medicine — Clinical Terms) [2], maintained by SNOMED International, is the most comprehensive international standard: over 350,000 active concepts organized in a polyhierarchical structure with explicit semantic relationships, more than 1,000,000 validated descriptions in multiple languages, and curated clinical text definitions. Its growing adoption in electronic health records makes automated understanding of its vocabulary a high-value practical capability.

Pre-trained LLMs trained on general corpora understand basic medical language but present significant limitations with SNOMED CT specialized terminology: they do not distinguish between admitted synonyms and canonical terms (FSN), do not handle semantic qualifiers in parentheses, and do not capture the is-a hierarchical relationships that structure the ontology. Domain-specific fine-tuning on this vocabulary overcomes these limitations efficiently through the PEFT (Parameter-Efficient Fine-Tuning) paradigm.

This work presents the end-to-end development of a SNOMED CT-specialized LLM: from corpus construction to local deployment executable on hardware without internet access. The objective is not code prediction—which belongs to retrieval systems—but deep vocabulary comprehension: synonyms, hierarchies, clinical definitions, and SNOMED model attributes. Direct applications include clinical text normalization, medical documentation assistants, and standardized terminology training.

II. THEORETICAL BACKGROUND

A. SNOMED CT and RF2 Files

SNOMED CT is distributed via Release Format 2 (RF2) files [2], a set of tab-delimited text files encoding ontological components. Files used in this project: sct2_Description.txt (concepts, FSNs and synonyms with active/inactive status), sct2_TextDefinition.txt (clinical text definitions), sct2_Relationship.txt (is-a relations and SNOMED model attributes), and sct2_Concept.txt (concept status). The Spanish edition was used, including FSN translations and synonyms validated by SNOMED International.

B. Quantized Low-Rank Adaptation (QLoRA)

QLoRA [3] combines two complementary techniques: 4-bit quantization of the base model (reducing ~14 GB to ~3.8 GB VRAM via NormalFloat NF4 format) and low-rank adapter training (LoRA [4]). LoRA decomposes weight updates as $\Delta W = BA$, where $B \in \mathbb{R}^{(d\times r)}$ and $A \in \mathbb{R}^{(r\times k)}$ with rank $r \ll \min(d,k)$. Only A and B are trained, reducing trainable parameters to 0.28% of the total in a 7-billion-parameter model with r=16 applied to seven

transformer modules. Base model weights remain frozen in 4-bit throughout training.

C. Mistral-7B-Instruct-v0.2

Mistral-7B-Instruct-v0.2 [5] was selected as base model for three characteristics: (a) Sliding Window Attention (SWA) architecture improving efficiency on long sequences; (b) Group-Query Attention (GQA) reducing KV cache usage; (c) superior benchmark performance in its size category. Its compatibility with 4-bit quantization and the Ollama ecosystem makes it suitable for local deployment.

D. Ollama — Local Inference

Ollama [6] is an open-source framework for running LLMs locally via a REST API compatible with the OpenAI standard. It supports the GGUF format (GPT-Generated Unified Format) for CPU and GPU quantization, and allows creating custom models via Modelfiles specifying the base model, adapters, and generation parameters. Integrating the fine-tuned model into Ollama enables inference without internet access or cloud service dependencies.

III. CORPUS DESIGN AND EVOLUTION

A. Corpus v1 — Code Prediction (Discarded)

The initial corpus formulated fine-tuning as prediction: given a clinical term, the model should predict the corresponding SNOMED CT numeric code. This formulation was discarded for two reasons: (a) codes are 8-18 digit identifiers with no learnable linguistic pattern; (b) grad_norm spikes repeated at exactly the same dataset position (epoch 0.154-0.160) across all sessions regardless of learning rate—an unequivocal signal of a data problem, not a hyperparameter problem.

B. Corpus v2 — Vocabulary and Equivalences

The redesign reoriented the objective: the model learns synonyms, equivalences, and definitions, not codes. Types 1 (canonical FSN definition with synonyms) and Type 2 (synonym-to-canonical-term equivalence) were introduced. With 50,000 examples, training achieved minimum loss 0.377 at step 1,250 and accuracy 91.3%. Inference tests revealed limitations in clinical text definitions and hierarchical relationships.

C. Corpus v3 and v4 — Additional Semantic Types

Type 3 incorporated clinical text definitions from sct2_TextDefinition.txt files, with an adaptive ratio filter based on the 95th length percentile to exclude atypical definitions. Type 4 incorporated is-a hierarchical chains. The recursive implementation of Type 4 caused a StackOverflow processing the 350,000+ nodes of the complete SNOMED CT hierarchy, resolved in the final version via completely iterative bottom-up BFS that processes each node only after all its parents have been processed.

D. Corpus v5_final — Complete Design (2,000,000 examples)

The final version incorporates five complementary semantic types. Type 5 adds SNOMED model concrete values (attributes with their values), completing the ontology representation. Script preprocess_snomed_v5_final.py generates the corpus from RF2 files via the bottom-up BFS algorithm, producing file snomed_ct_corpus_v5_final.jsonl. These results correspond to the initial tests; the complete fine-tuning on the v5_final corpus (2M examples) is currently underway with an estimated 60 days remaining.

Type	Semantic description	RF2 Source
Type 1	Canonical definition: FSN + active synonym list with status	sct2_Description.txt
Type 2	Equivalence: synonym → FSN with parenthetical qualifiers	sct2_Description.txt
Type 3	Clinical text definition with adaptive ratio filter (P95)	sct2_TextDefinition.txt
Type 4	Hierarchical chain BFS bottom-up (up to 3 chains per concept)	sct2_Relationship.txt
Type 5	SNOMED model attributes with concrete values	sct2_Relationship.txt

Table I — Example types in corpus v5_final

E. Corpus Statistical Analysis

A statistical analysis was performed on all 2,000,000 examples to evaluate whether length distribution could cause training instabilities:

Metric	Value	Comparative reference
Typedistribution	100% uniform	No concentration of any type
Mean length	67.8 tokens	Appropriateformax_length=256
Maximum length	293 tokens	Outlier; truncatedby script

95th percentile (P95)	121 tokens	1.78× the mean — lowvariance
P95/mean ratio	1.78×	Wikipedia: 1.5-2.0 / Common Crawl: 1.8-2.2

Table II — Corpus v5_final statistics (2,000,000 examples)

The P95/mean ratio of 1.78 confirms that observed instabilities are not attributable to the corpus but exclusively to the optimizer-scheduler incompatibility mechanism described in Section V.

IV. FINE-TUNING CONFIGURATION

A. Hardware Environment and Constraints

Training was conducted on an NVIDIA Turing GPU (CC 7.5) with 6 GB VRAM under Windows 10/11, Python 3.11 in an Anaconda environment. The most critical constraint was bfloat16 incompatibility on CC 7.5: Accelerate's Grad Scaler automatically activates bfloat16, producing a runtime error. The solution was to force float32 (fp32) for all operations, which doubles activation memory consumption compared to fp16 and makes max_length=256 the safe practical limit with 6 GB VRAM.

B. Implementation Errors and Solutions

The setup process required resolving fourteen documented implementation errors. The most significant were: (a) renaming max_seq_length → max_length and tokenizer → processing_class in TRL ≥1.0 API; (b) automatic gradient_checkpointing activation by default in TRL ≥1.0, incompatible with prepare_model_for_kbit_training(); (c) requirement of use_reentrant=False in PyTorch 2.9+ for torch.utils.checkpoint; (d) paged_adamw_8bit incompatibility with Windows that incorrectly activates GradScaler.

C. Final Hyperparameters

Parameter	Final value	Justification / evolution
Base model	Mistral-7B-Instruct-v0.2	SWA+GQA architecture; GGUF/Ollama compatible
Quantization	4-bit NF4	Reduces 14 GB → 3.8 GB VRAM
LoRA r / α	16 / 32	Standard for 7B domain fine-tuning
Target modules	7 (q,k,v,o,gate,up,downproj)	Full coverage of attention and MLP layers
learning_rate (η_{max})	4×10^{-5}	Reducedfrom 2×10^{-4} → 8×10^{-5} → 4×10^{-5}
lr_scheduler	Cosinewith linear warmup	Transformer standard; warmup=3% of total
max_grad_norm	0.1	Reduced from 0.3 → 0.1 after sustained spikes
Optimizer	AdamW (adamw_torch)	paged_adamw_8bit incompatible with Windows
Effectivebatch	32 (batch=1, accum.=32)	Compensates 1 example/step limit on 6 GB
max_length	256 tokens	Safe limit in fp32 with 6 GB VRAM
Steps per epoch	43,750	1,400,000 examples / effectivebatch 32
save_steps / limit	200 / 3	Checkpoint every 200 steps; retains 3 most recent

Table III — Final training hyperparameters

D. First Complete Epoch Metrics

Training on corpus v2 (44,000 examples) completed the first epoch in approximately 29 hours. Post-warmup metrics: loss mean $\mu=0.385$ ($\sigma=0.013$) with normal distribution; mean accuracy 91.1%; 17 grad_norm spikes >5 (1.1% of steps), all with immediate recovery in the following step; loss-accuracy correlation $r=-0.89$. Accuracy exceeded 91.5% at approximately epoch 50% and remained above it for the rest of training.

V. INSTABILITY MANAGEMENT AND THE CSOR FRAMEWORK

A. Checkpoint Resumption Instability

When resuming training from a checkpoint with a modified peak learning rate, a systematic failure mechanism with two components was identified. The first is scheduler.pt precedence: this file stores η_{max} in its base_lrs attribute, and upon resumption the HuggingFace Trainer calls load_state_dict() before any training step, overriding the learning_rate parameter in the training script. Changing the

learning rate in the script without deleting scheduler.pt has no effect. This precedence was confirmed empirically: in two separate runs (R1 with $\eta_{max}=8\times10^{-5}$ in the script and R2 with $\eta_{max}=4\times10^{-5}$ in the script), logs showed learning_rate=5.50×10^{-5} at step 1,410 in both—and both collapsed at the same step with identical curves.

The second component is AdamW [7] momentum mismatch. The second moment v_t, accumulated with $v(t_0) \propto \eta^2_{max,old}$, produces disproportionate updates when warmup restarts at $\eta\approx0$: the denominator $\sqrt{\hat{v}}_t$ remains large while the numerator $\eta_k \cdot \hat{m}_t$grows, eventually exceeding it with time constant $\tau=1/(1-\beta_2)=1{,}000$ steps. In fp32 without GradScaler, this culminates in numerical overflow (observed loss: 2.15×10^{8}).

B. Cold-Start Optimizer Resumption (CSOR)

The CSOR framework [1] proposes the deliberate deletion of scheduler.pt and optimizer.pt from the checkpoint directory before resuming, preserving adapter_model.safetensors. This initializes the scheduler with η_max,new from the current script and the optimizer with $m_t=0$, $v_t=0$, eliminating the mismatch without losing accumulated knowledge in the LoRA weights. The cost is a recalibration period of 200-400 steps (<0.5% of total epoch steps).

CSOR is theoretically grounded in two precedents: (a) SGDR warm restarts [8], which propose periodically resetting the scheduler to escape local minima; (b) the standard practice of initializing the optimizer from scratch in transfer learning [9], where the pre-training optimizer state is discarded when adapting to a new task.

Run	CKP	sched.pt	optim.pt	η_{max}effective	Outcome
R1	1,400	preserved	preserved	8×10^{-5}	NaN overflow step ~1,650. Loss: 2.15×10^{8}
R2	1,400	preserved	preserved	8×10^{-5} (script ignored)	NaN overflow step ~1,650. Precedence confirmed
R3	1,400	deleted	preserved	4×10^{-5}	Deterioration: loss MA 0.421→0.436→0.480
R4 (CSOR)	1,400	deleted	deleted	4×10^{-5}	Stable. Min loss: 0.370. Max acc: 91.5%
R5 (CSOR)	1,400	deleted	deleted	4×10^{-5}	Stable to step 2,600
R6 (CSOR)	2,600	deleted	deleted	4×10^{-5}	Ongoing — first 24 stepsstable

Table IV — Six resumptions: conditions and outcomes

C. Quantitative CSOR Validation

Metric	R3 (scheduler.pt only deleted)	R4 — CSOR (bothdeleted)
Mean grad_norm	12.4	1.9
Maximum grad_norm	210.5	7.09
Stepswithgrad_norm>20	23/400 (57.5%)	0/400 (0%)
Loss at step +400	0.526 (deteriorating)	0.370 (new historicalminimum)
Accuracy at step +400	88.2% (declining)	91.5% (new historicalmaximum)

Table V — Comparative metrics: R3 vs R4-CSOR (first 400 post-resumption steps)

VI. LOCAL DEPLOYMENT WITH OLLAMA

A. Conversionto GGUF Format

The LoRA adapters generated by fine-tuning (adapter_model.safetensors, ~165 MB) are merged with the quantized base model to produce a complete model in GGUF format, compatible with Ollama. The merge process has two stages: (a) LoRA adapter merge onto the fp32 base model using PEFT's merge_and_unload(), obtaining a complete 7,240M-parameter fp32 model; (b) quantization to GGUF Q4_K_M format via llama.cpp, reducing from ~28 GB to ~4.5 GB while maintaining comparable inference quality.

B. OllamaModelfile Configuration

Ollama uses a Modelfile to define the custom model, system prompt, and generation parameters. The SNOMED CT specialized model Modelfile specifies: FROM with the path to the GGUF file, SYSTEM

with behavioral instructions (respond to SNOMED CT terminology queries in Spanish), and PARAMETER for temperature (0.1 for deterministic responses), top_p, and num_ctx (2048 token context).

C. Local Inference

Once registered in Ollama via ollama create snomed-ct-es -f Modelfile, the model is locally accessible through the REST API on port 11434, compatible with standard OpenAI clients. CPU inference requires approximately 8-12 GB RAM; on CUDA-capable GPU, response time reduces to 2-5 seconds for typical clinical terminology queries.

VII. RESULTS, CONCLUSIONS, AND FUTURE WORK

A. Main Results

The project has completed the following stages: (1) construction of the SNOMED CT corpus of 2,000,000 examples across five semantic types with a statistically uniform distribution; (2) initial fine-tuning tests on a reduced subset of 50,000 examples, achieving a minimum loss of 0.370 and token accuracy of 91.5%, surpassing the preliminary results obtained with previous configurations; (3) functional local deployment using Ollama. The CSOR framework, formulated during the process, resolved the instability problem in checkpoint resumptions, eliminating grad_norm spikes >20 in the first 400 steps after resumption. The full-scale training on the v5_final corpus is currently in progress, with an estimated remaining duration of 60 days. The final results will be reported in a subsequent publication.

B. Limitations

Main limitations are: (a) hardware restricted to fp32 without GradScaler, amplifying difficult-batch impact and making training more susceptible to instabilities than in bf16 or fp16 configurations; (b) evaluation limited to training metrics, without formal inference benchmarks on a complete test corpus; (c) Spanish-language corpus coverage; extension to other languages requires the corresponding language RF2 files. The quantitative evaluation presented in this article is based exclusively on the reduced corpus of 50,000 examples; the final metrics on the full corpus may vary.

C. Future Work

Natural project extensions include: (a) formal evaluation via clinical normalization benchmarks (MedMentions, CLEF eHealth); (b) integration of the specialized LLM as normalization component in a RAG architecture with SNOMED CT vector database, where the LLM processes free text and the retrieval system resolves code assignment; (c) extension of the corpus to the international English edition of SNOMED CT; (d) experimentation with larger base models (13B, 34B) once bf16-compatible hardware is available.

D. Conclusions

This work demonstrates the feasibility of specialized fine-tuning of LLMs on structured clinical terminology using consumer-grade hardware, even at a preliminary execution stage. The main contributions are: (1) the design and construction of the v5_final corpus with statistical analysis confirming its distributional quality; (2) the systematic documentation of implementation errors in QLoRA on constrained hardware with verified solutions; (3) the formulation and empirical validation of the CSOR framework, which in initial tests reduced the mean grad_norm from 12.4 to 1.9 and completely eliminated spikes >20 over 400 steps; (4) local deployment of the model in the Ollama ecosystem. The complete fine-tuning process on the 2 million examples of the v5_final corpus is currently underway, with an estimated 60 days of computation remaining. The results reported here (loss 0.370, accuracy 91.5%) correspond to the initial tests with a subset of 50,000 examples and confirm the validity of the methodology.

GLOSSARY

AdamW: Adam optimizer variant with decoupled weight decay regularization. Standard for transformer training.

BFS (Breadth-First Search): Graph search algorithm processing all nodes at one level before advancing to the next. Used to traverse the SNOMED CT hierarchy without stack overflow.

CSOR (Cold-Start Optimizer Resumption): Framework proposed in this work: delete scheduler.pt and optimizer.pt before resuming from a checkpoint to avoid incompatibility between accumulated optimizer state and the restarted scheduler.

Fine-tuning: Adjustment of a pre-trained model on domain-specific data, modifying a subset of parameters to specialize its behavior.

FSN (Fully Specified Name): Fully specified name of a SNOMED CT concept. Includes the canonical term plus a semantic qualifier in parentheses (e.g., 'Type 2 diabetes mellitus (disorder)').

GGUF: File format (GPT-Generated Unified Format) for efficient storage of quantized language models. Used by llama.cpp and Ollama for local inference.

Grad_norm: Gradient norm: total magnitude of all gradients before being applied to the optimizer. High values indicate potentially unstable updates.

Learning rate (LR): Size of the weight update step at each iteration. Critical training parameter.

LoRA (Low-Rank Adaptation): PEFT technique inserting trainable low-rank matrices into model layers. Only these matrices are updated, keeping the rest of the model frozen.

LLM (Large Language Model): Transformer-architecture neural network trained on large text volumes to understand and generate natural language.
NF4 (NormalFloat 4-bit): 4-bit quantization format designed for normally distributed data, used in QLoRA to compress base model weights.
Ollama: Open-source framework for running language models locally, with support for GGUF format models and an OpenAI-compatible REST API.
PEFT (Parameter-Efficient Fine-Tuning): Family of techniques that adapt pre-trained models by modifying only a small fraction of parameters, drastically reducing computational requirements.
QLoRA: Quantized LoRA: combines 4-bit quantization of the base model with trainable LoRA adapters for efficient fine-tuning on hardware with limited VRAM.
RF2 (Release Format 2): SNOMED CT distribution format via tab-delimited text files encoding concepts, descriptions, relationships, and other ontological components.
scheduler.pt: Learning rate scheduler state file saved at each checkpoint. Contains η_max (base_lrs) and the current schedule position, and takes precedence over script parameters when resuming training.
SNOMED CT: Systematized Nomenclature of Medicine — Clinical Terms. International clinical terminology standard with over 350,000 active concepts organized in a polyhierarchical structure.
Warmup: Initial training phase in which the learning rate gradually rises from 0 to its maximum value to avoid abrupt updates when the model is not yet oriented.
VRAM: GPU video memory. The limiting factor in LLM fine-tuning on consumer hardware.

REFERENCES

[1] Antokoletz Huerta, D., & La Rosa, F. M. (2026). *Cold-Start Optimizer Resumption in QLoRA Fine-Tuning: Theoretical Foundations, Empirical Validation, and Practical Implications.* Universidad de Morón.

[2] SNOMED International. (2024). *SNOMED CT Release Format 2 (RF2) Specification.*.https://confluence.ihtsdotools.org/display/DOCRELFMT

[3] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). *QLoRA: Efficient finetuning of quantized LLMs.* Advances in Neural Information Processing Systems (NeurIPS 2023). https://doi.org/10.48550/arXiv.2305.14314

[4] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). *LoRA: Low-rank adaptation of large language models.* International Conference on Learning Representations (ICLR 2022). https://doi.org/10.48550/arXiv.2106.09685

[5] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., & El Sayed, W. (2023). *Mistral 7B.*arXiv preprint. https://doi.org/10.48550/arXiv.2310.06825

[6] Ollama. (2024). *Ollama: Get up and running with large language models.*. Open-source software. https://ollama.com

[7] Loshchilov, I., & Hutter, F. (2019). *Decoupled weight decay regularization.* International Conference on Learning Representations (ICLR 2019). https://doi.org/10.48550/arXiv.1711.05101

[8] Loshchilov, I., & Hutter, F. (2017). *SGDR: Stochastic gradient descent with warm restarts.* International Conference on Learning Representations (ICLR 2017). https://doi.org/10.48550/arXiv.1608.03983

[9] Howard, J., & Ruder, S. (2018). *Universal language model fine-tuning for text classification.* Proceedings of the 56th Annual Meeting of the ACL (ACL 2018). https://doi.org/10.18653/v1/P18-1031

[10] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & Rush, A. M. (2020). *Transformers: State-of-the-art natural language processing.* Proceedings of EMNLP 2020: System Demonstrations. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[11] Gao, T., Fisch, A., & Chen, D. (2021). *Making pre-trained language models better few-shot learners.* Proceedings of ACL-IJCNLP 2021. https://doi.org/10.18653/v1/2021.acl-long.295

[12] Kraljevic, Z., Searle, T., Shek, A., Roguski, L., Noor, K., Bean, D., Mascio, A., Zhu, L., Folarin, A. A., Roberts, A., Bendayan, R., Richardson, M. P., Stewart, R., Shah, A. D., Wong, W. K., Ibrahim, Z., Teo, J. T., & Dobson, R. J. B. (2021). *Multi-domain clinical natural language processing with MedCAT.* Artificial Intelligence in Medicine, 117, 102083. https://doi.org/10.1016/j.artmed.2021.102083

★★★