Stochastic Gradient Descenders at DeSegMa-IT: Instruction-Tuned LLM and Token Classification for MGT Detection and Boundary Localization

Huynh Nguyen Phu[1,2], Bui Hong Son[1,2] and Dang Van Thin[1,2]

[1]University of Information Technology, Ho Chi Minh City, Vietnam
[2]Vietnam National University, Ho Chi Minh City, Vietnam

Abstract

We describe our top-ranked system for the DeSegMa-IT shared task (EVALITA 2026), which achieved **1st place** in Subtask B (boundary localization) with an MAE of 52.54. Our key insight is to reframe boundary detection as a *token classification* problem rather than direct character-offset regression: by labeling each token as human- or machine-written, we enable a monolingual Italian encoder (UmBERTo) to capture fine-grained stylistic shifts and pinpoint the transition point. Unlike sequence-level classifiers or generative approaches that predict offsets end-to-end, our token-level formulation provides richer supervision and naturally handles variable-length documents. For Subtask A (binary classification), we fine-tuned Qwen2.5-0.5B-Instruct with LoRA, achieving Rank 5 (Accuracy 0.92).

1. Introduction

The proliferation of Large Language Models (LLMs) capable of generating high-quality text has created an urgent need for robust detection systems. This is particularly critical in scenarios involving distribution shifts, where detectors trained on one domain must generalize to unseen topics or prompting strategies. DeSegMa-IT [1], a shared task at EVALITA 2026 [2], addresses these challenges by benchmarking systems on two complementary problems: document-level detection of machine-generated text (MGT) and fine-grained segmentation of human-machine mixed content.

The shared task is divided into two subtasks:

- **Subtask A (Binary Classification):** Determining whether a full document was written by a human or generated by a machine.
- **Subtask B (Boundary Localization):** Identifying the exact character index where a human-written prefix transitions into a machine-generated continuation.

Our work investigates two key research questions guiding our dual-system design: (1) Can reformulating boundary localization as token classification—rather than direct offset regression—yield more accurate transition-point detection by leveraging fine-grained stylistic signals? (2) How effectively can parameter-efficient fine-tuning of a lightweight instruction-tuned LLM generalize to unseen MGT scenarios with limited computational resources? To address these, we make three main contributions:

- **Reformulate** the boundary localization task as dense token classification, enabling token-level supervision that captures local stylistic shifts between human and machine text.
- **Achieve state-of-the-art** performance on Subtask B (**Rank 1**, MAE 52.54), demonstrating that token-level formulation outperforms regression-based and sequence-level baselines.
- **Demonstrate** competitive binary classification (Rank 5, Accuracy 0.92) using LoRA-tuned Qwen2.5-0.5B-Instruct with only 0.5B parameters—an order of magnitude smaller than typical LLM-based detectors.

EVALITA 2026: 9th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, Feb 26–27, Bari, IT
24521352@gm.uit.edu.vn (H. N. Phu); 22521246@gm.uit.edu.vn (B. H. Son); thindv@uit.edu.vn (D. V. Thin)
0009-0004-6867-5853 (H. N. Phu); 0009-0006-7420-9212 (B. H. Son); 0000-0001-8340-1405 (D. V. Thin)

Table 1
Dataset statistics for both subtasks.

Subtask	Metric	Train	Dev	Test
A	Total samples	33,138	5,059	15,135
	Human (label=0)	16,569	—	—
	Machine (label=1)	16,569	—	—
B	Total samples	19,945	4,579	26,015

- **Provide** empirical analysis of error patterns and boundary-offset distributions, offering insights for future MGT detection research.

In this paper, we describe the system submitted by the *Stochastic Gradient Descenders* team. Our approach prioritizes specialized architectural choices for each subtask rather than a monolithic solution. For Subtask A, we employ a prompt-based learning strategy, fine-tuning a lightweight instruction-following model (`Qwen2.5-0.5B-Instruct` [3]) via Low-Rank Adaptation (LoRA) [4]. This allows us to leverage the reasoning capabilities of LLMs while maintaining high computational efficiency. For Subtask B, we treat segmentation as a dense token-labeling problem, utilizing a fully fine-tuned Italian encoder (UmBERTo [5]) to capture local stylistic shifts.

The remainder of this paper details our methodology (Section 3), experimental setup (Section 4), and a discussion of the results and error analysis (Section 6).

2. Task and Dataset Description

DeSegMa-IT [1] is a shared task at EVALITA 2026 that benchmarks machine-generated text (MGT) detection systems on Italian text.

2.1. Task Definitions

The shared task comprises two subtasks:

- **Subtask A (Binary Classification):** Given a document, predict whether it is entirely human-written (label 0) or machine-generated (label 1). Evaluation metric: Accuracy.
- **Subtask B (Boundary Localization):** Given a mixed document (human prefix + LLM continuation), predict the 0-based character index where the transition occurs. Evaluation metric: Mean Absolute Error (MAE).

2.2. Dataset Statistics

Table 1 summarizes the dataset splits for both subtasks. Subtask A contains 33,138 training samples with a perfectly balanced label distribution (50% human, 50% machine). Subtask B provides 19,945 training samples of human–machine hybrid documents.

2.3. Text Length Analysis

Subtask A. Documents exhibit high variance in length: mean 2,423 characters (median 1,697, 95th percentile 5,525). This motivates our use of truncation-aware prompting for the LLM-based classifier.

Subtask B. Mixed documents are relatively uniform in length (mean 1,416, std 302 characters). The boundary position (`human_len`) averages 265 characters with a mean ratio of 0.20 relative to document length, indicating that most transitions occur in the first quarter of the text. This skewed distribution informed our token-level labeling strategy, which can capture fine-grained transitions regardless of absolute position.

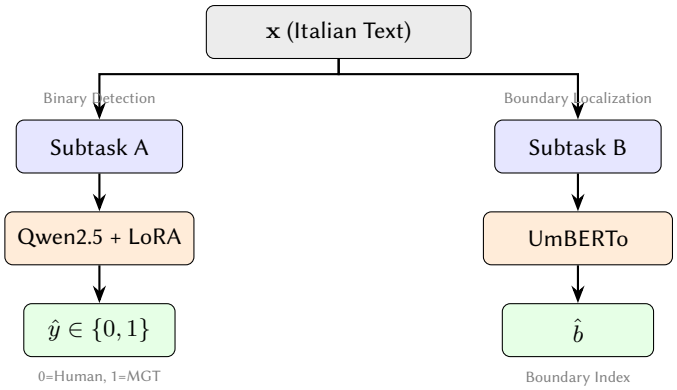

Figure 1: System overview. Our approach employs two independent pipelines: an instruction-tuned LLM for binary detection (Subtask A) and a token classifier for boundary localization (Subtask B).

3. Methodology

This section details the system architecture and training pipelines employed for the DeSegMa-IT shared task. We present our approaches for Subtask A and Subtask B sequentially, as illustrated in Figure 1.

3.1. Subtask A: Binary Detection via Instruction Tuning

Problem Formulation

We approach Subtask A as a conditional generation problem using a decoder-only Large Language Model (LLM). Rather than initializing a randomly weighted classification head on top of the backbone, we leverage the model's pre-existing instruction-following capabilities. The task is framed as a single-token generation constraint: the model is prompted to output a specific digit representing the class label (see Figure 2).

Data Formatting and Supervision

For each input document, we construct a conversation-style prompt adhering to the *ChatML* format. The system instruction is consistent across all instances:

> *"You are a binary MGT detector. Return exactly one digit: 0 for human-written, 1 for machine-generated."*

The user message contains the raw document text. During training, the assistant's response is the ground-truth label ("0" or "1"). We employ standard supervised fine-tuning (SFT) with separate masking: the cross-entropy loss is computed exclusively on the assistant's response tokens, while the system and user instructions are masked out.

Backbone and Parameter-Efficient Adaptation

Our training infrastructure for Subtask A (2× NVIDIA T4 on Kaggle) constrains the backbone to models under 3B parameters. Within this budget, we experimented with two sizes from the `Qwen2.5-Instruct` family [3]: the 0.5B and 3B variants. In preliminary trials, the 0.5B model achieved higher validation accuracy than its 3B counterpart, which we attribute to the larger model's tendency to overfit under LoRA fine-tuning with limited training epochs. We therefore select `Qwen2.5-0.5B-Instruct` as our backbone, which also benefits from strong multilingual coverage (including Italian) and robust instruction-following abilities at this compact scale. To mitigate catastrophic forgetting and reduce computational overhead, we apply Low-Rank Adaptation (LoRA) [4] using the `unsloth` library for optimized training [6]. Our LoRA configuration targets all linear layers with rank $r = 16$, $\alpha = 32$, and dropout rate 0.05. We train with 16-bit precision (mixed precision) to preserve dynamic range, avoiding 4-bit quantization to maximize performance given the manageable model size.

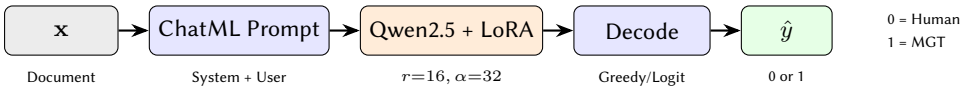

Figure 2: Instruction-tuning pipeline for Subtask A. The document is formatted as a ChatML prompt, processed by Qwen2.5 with LoRA adaptation, and decoded to predict the binary label $\hat{y}$.

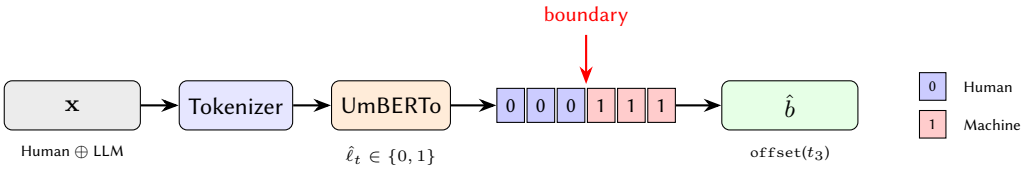

Figure 3: Token classification pipeline for Subtask B. Each token receives a binary label, and the boundary $\hat{b}$ is the offset of the first machine-labeled token.

Training Protocol

We allocate an 80/20 stratified split of the provided training set for internal validation. Training is performed for 2 epochs with a maximum sequence length of 1024 tokens. We use a per-device batch size of 4 with gradient accumulation steps set to 5, resulting in an effective batch size of 20. The optimizer is AdamW [7] (8-bit quantization [8]) with a learning rate of 1×10^{-4}, a warmup ratio of 0.05, and a weight decay of 0.01. We disable the `use_cache` configuration to support gradient checkpointing during training.

Inference Strategy

At inference, we set `max_new_tokens=2` as a safety buffer: the model is expected to produce a single digit followed by the EOS token, but occasionally emits a whitespace or newline character before the digit. Allowing two tokens ensures the digit is captured even when preceded by such spurious characters. The generated string is stripped of whitespace and only its first character is inspected, so the second token slot is never used for classification. To ensure robustness, we implement a two-step fallback mechanism:

1. **Greedy Decoding:** The stripped output is checked; if its first character is "0" or "1", it is accepted as the prediction.
2. **Logit Fallback:** In rare cases where neither character appears (e.g., the model produces only special tokens), we perform a separate forward pass on the original prompt and inspect the logits at the *last prompt position*—i.e., the distribution over the *first* token the model would generate. The class whose token ID receives the higher logit ("0" vs. "1") is selected.

3.2. Subtask B: Boundary Localization via Token Classification

Token-Level Formulation

While Subtask B is evaluated as a regression task (predicting a character index), we reformulate it as a dense sequence labeling problem. Given a document x composed of a human-written prefix and a machine-generated continuation, we tokenize the sequence and assign a binary label ℓ_t to each token t:

$$\ell_t = \begin{cases} 0 & \text{if start\_offset}(t) < \text{boundary\_index} \\ 1 & \text{otherwise} \end{cases}$$

This simple rule assigns a token to the human class if it *begins* before the boundary, effectively treating tokens that straddle the boundary as human. The formulation provides dense supervision signals compared to a single scalar regression target, allowing the model to learn local textural shifts (see Figure 3).

Backbone Model
We fine-tune `Musixmatch/umberto-commoncrawl-cased-v1` (UmBERTo) [5], a RoBERTa-based [9] model pre-trained on a large Italian corpus. A standard token classification head is initialized on top of the encoder to project hidden states into the binary label space.

Training Protocol
The official training data is split 80/20 for training and validation. We tokenize inputs with a maximum length of 512 tokens using truncation. Crucially, we maintain full offset mappings to allow precise reconstruction of character indices during inference. The model is fully fine-tuned (all parameters) for 10 epochs using mixed precision (FP16). We use a batch size of 8 per device with gradient accumulation 2 (effective batch size 16), a learning rate of 2×10^{-5}, linear warmup of 0.1, and weight decay of 0.01. Evaluation is performed every 250 steps to monitor the Mean Absolute Error (MAE) on the validation set.

Decoding Strategy
Inference involves a two-step process:

1. **Prediction:** The model outputs per-token class probabilities. We take the argmax to obtain a sequence of binary labels.
2. **Boundary Extraction:** We iterate through the predicted labels to find the first token classified as 'machine' ($\hat{\ell}_t = 1$). The start character offset of this token is returned as the predicted boundary. If no token is classified as machine, the boundary is set to the document length.

This strategy aligns with the task definition (identifying the *start* of the generated text) while leveraging the contextual power of the bidirectional encoder.

4. Experimental Setup

4.1. Implementation Details

All experiments were implemented using the PyTorch framework [10] and the Hugging Face ecosystem [11]. For Subtask A, we utilized the `unsloth` library [6] to accelerate the LoRA fine-tuning of Qwen2.5, trained on 2× NVIDIA T4 GPUs provided by Kaggle.[1] For Subtask B, we used the standard `Trainer` API from the Transformers library, trained mainly on a single NVIDIA H100 (80GB VRAM) provided by FPT AI Factory.[2] To ensure reproducibility, we fixed the random seed to 42 for all data splits and model initializations.

4.2. Validation Protocols

Subtask A We employed a stratified hold-out validation strategy, reserving 20% of the provided training data. Model checkpoints were evaluated based on classification accuracy and the formatting validity of the generated tokens. Given the stability of the instruction-tuned backbone, we observed rapid convergence; the final submission utilized the checkpoint from the end of the second epoch.

Subtask B For the segmentation task, we used an 80/20 random split of the labeled data. Metrics were computed at the character level (MAE) rather than the token level to align with the official competition metric. During development, we also explored a 5-fold cross-validation scheme to assess the robustness of different encoder backbones before selecting UmBERTo for the final submission.

[1] https://www.kaggle.com
[2] https://ai.fptcloud.com/

Table 2
Offline validation results for Subtask B. The *Formulation* column refers to the decoding heuristic used.

Backbone	Formulation	MAE
Italian BERT (5-fold CV)	Token Cls + First Offset	55.78
mDeBERTa-v3	Token Cls + First Offset	55.35
UmBERTo	Token Cls + First Offset	53.30
UmBERTo	Token Cls + Global+Biased	52.06

Table 3
Results of progressive method improvements on the development set for Subtask B. Each row adds one component to the previous configuration. Δ indicates the improvement over the previous row.

#	Configuration	MAE	Δ
1	UmBERTo baseline (3 epochs, simple decode)	55.12	—
2	+ Increase to 5 epochs	53.30	-1.82
3	+ Global score decoding	52.89	-0.41
4	+ Length bias tuning ($\beta = -0.15$)	52.06	-0.83
Final system (row 4)		**52.06**	—

4.3. Baseline Comparisons for Subtask B

In the preliminary phase, we benchmarked several architectures for the boundary localization task:

- **mDeBERTa-v3-base** [12]: A multilingual transformer with 86M backbone parameters (276M total with embeddings), featuring disentangled attention that separately encodes content and position. The model was pre-trained on the CC100 multilingual corpus with a 250K-token vocabulary.
- **Italian BERT** (`bert-base-italian-xxl-cased`) [13]: A monolingual BERT variant with approximately 110M parameters, pre-trained on OPUS corpora extended with the Italian portion of OSCAR (81GB total). The cased tokenization preserves orthographic cues relevant for stylistic analysis.
- **UmBERTo** (`umberto-commoncrawl-cased-v1`) [5]: A RoBERTa-based Italian model employing SentencePiece tokenization with a 32K vocabulary and whole-word masking. Pre-trained on the Italian subcorpus of OSCAR, it captures language-specific morphological and syntactic patterns.

All baselines were trained with the same token-classification objective described in Section 3.

4.4. Offline Validation Results

Table 2 presents the offline performance of our Subtask B models on the development splits. UmBERTo consistently outperformed other backbones, likely due to its superior handling of Italian morphology and syntax compared to the multilingual or standard BERT variants. We also experimented with a "global + biased" decoding strategy (tuning a bias term to penalize/reward the positive class), which yielded further improvements in offline settings. However, for the final submission, we opted for the simpler standard decoding to avoid overfitting to the development set distribution. To understand the contribution of each component, we also conducted ablation experiments starting from a vanilla UmBERTo baseline and progressively adding components (Table 3).

We also experimented with several training-time tricks that ultimately did *not* improve the final performance:

Key Findings:

Table 4
Negative results: training tricks that degraded or did not improve performance on the development set.

Configuration	MAE	vs. Baseline
Weighted sampling (extreme cases $\times 3.0$)	56.87	$+1.75$
Boundary token re-weighting (k=5)	54.21	-0.91
Custom regression head (λ=0.1)	55.43	$+0.31$
Custom regression head (λ=0.01)	53.67	-1.45
Boundary weighting + regression (k=3, λ=0.01)	53.95	-1.17

Table 5
Official Subtask A leaderboard [14].

Pos.	Team	Accuracy
1	Gradient Descenders	0.945 755
2	Kenji Endo	0.942 649
3	UniTor	0.928 774
4	Nicla	0.924 348
5	**Stochastic Gradient Descenders (Ours)**	0.921 638

- **Training epochs:** Extending training from 3 to 5 epochs provided consistent gains, though validation loss plateaued after epoch 5–6.
- **Decoding strategy:** The global score decoding, which considers the joint probability of all tokens rather than the first positive prediction, improved boundary precision.
- **Length bias:** Tuning a small bias term ($\beta = -0.15$) to slightly penalize the machine-generated class probability helped correct for systematic over-prediction of short boundaries.
- **Weighted sampling:** Oversampling extreme boundary lengths (very short or very long human segments) caused over-prediction bias, likely because the model learned to expect extreme cases more frequently than in the test distribution.
- **Auxiliary regression head:** Adding a regression head to directly predict the boundary character index did not consistently help, suggesting that the token classification objective already captures sufficient information.

5. Official Results

We report the official leaderboard results as published by the organizers [14]. Following the paper structure, Subtask A results are presented before Subtask B.

5.1. Subtask A: Binary Classification

Table 5 presents the official Subtask A leaderboard, where our system achieves 5th place with an accuracy of 0.9216, trailing the top-performing team (Gradient Descenders) by 2.4 percentage points. The result demonstrates that instruction-tuned LLMs can achieve competitive classification performance with minimal task-specific adaptation. By applying LoRA to Qwen2.5-0.5B-Instruct, we update only a small fraction of the model's parameters while preserving its pre-trained reasoning capabilities, yet reach accuracy within a reasonable margin of the top-ranked systems. Notably, the top-4 systems cluster within a narrow 2.1-point accuracy band (0.9246–0.9458), which may suggest that the remaining performance gap is increasingly driven by stylistic ambiguity in edge cases rather than by differences in model capacity alone.

Table 6
Official Subtask B leaderboard [14].

Pos.	Team	MAE
2	MINDS	56.53
3	Gradient Descenders	62.66
4	UniTor	81.60
5	Nicla	102.04
1	**Stochastic Gradient Descenders (Ours)**	52.54

5.2. Subtask B: Boundary Localization

Table 6 presents the official Subtask B leaderboard, where our system achieves **1st place** with an MAE of 52.54 characters—outperforming the runner-up (MINDS) by 3.99 points and the third-place team (Gradient Descenders) by a substantial 10.12 points (16.2% relative improvement). We attribute our strong performance primarily to two design decisions validated in our offline ablations (Section 4): (1) the choice of UmBERTo, a monolingual Italian encoder whose language-specific pre-training yields more reliable token-level predictions than multilingual alternatives, and (2) the simple first-positive decoding heuristic, which avoids introducing additional error through complex post-processing. Furthermore, the steep performance drop-off beyond the top-2 teams (UniTor at 81.60 MAE, Nicla at 102.04 MAE) indicates that boundary localization is a substantially harder task than binary classification, rewarding architectures specifically optimized for fine-grained sequence labeling over general-purpose approaches.

5.3. Summary

The proposed dual-strategy approach achieves competitive performance on Subtask A (5th place, 0.9216 accuracy) and state-of-the-art performance on Subtask B (1st place, 52.54 MAE). These results validate two complementary insights: (1) parameter-efficient LLM adaptation provides a strong baseline for binary detection without extensive fine-tuning, and (2) monolingual encoders with dense token-level supervision excel at fine-grained boundary localization. The contrasting outcomes across subtasks—mid-tier classification but top-tier segmentation—highlight that task-specific architectural choices matter more than raw model scale, particularly for structured prediction problems where language-specific morphological cues provide discriminative signal.

6. Discussion

This section presents a structured error analysis of our systems, identifying failure patterns and linking them to design choices. We support our analysis with confusion matrices and representative error cases.

6.1. Subtask A

Table 7 presents the confusion matrix on our 20% held-out validation set (6,628 samples). The system achieves high overall accuracy (92.2%), but the error distribution reveals asymmetric failure modes. From manual inspection of 100 randomly sampled errors, we identify three dominant patterns: false positives (8.2%) where human-written texts are misclassified as machine-generated, primarily when the writing is highly formal or follows templated structures such as press releases and legal notices; false negatives (7.4%) where machine-generated texts exhibiting high fluency and contextual coherence—particularly advanced LLM outputs mimicking journalistic style—are misclassified as human; and length-induced truncation errors, where the model only observes the initial 1,024 tokens and misses telltale machine artifacts in the continuation.

Two representative cases illustrate these failure modes. The first is a human-written corporate announcement—*"Il Consiglio di Amministrazione, riunitosi in data odierna, ha deliberato l'approvazione*

Table 7
Confusion matrix for Subtask A. Rows = ground truth, columns = predictions.

	Pred: Human	**Pred: Machine**	**Total**
Actual: Human	3,042	272	3,314
Actual: Machine	245	3,069	3,314
Total	3,287	3,341	6,628

Table 8
Token-level confusion matrix for Subtask B. Special tokens excluded.

	Pred: Human	**Pred: Machine**
Actual: Human	876,421	47,892
Actual: Machine	89,341	1,289,102

Table 9
Error analysis by human-prefix length.

Human Prefix	Count	MAE	Bias	Error %
0–100 chars	2,404	74.02	+73.17	9.3%
100–200 chars	6,345	57.13	+44.32	22.1%
200–300 chars	6,394	51.96	+18.14	14.6%
300–400 chars	6,369	47.70	−10.71	12.3%
400–512 chars	4,487	61.92	−52.97	17.8%

del bilancio consolidato per l'esercizio 2025…"—misclassified as machine-generated due to its bureaucratic register, passive constructions, and absence of subjective markers. The second is an LLM-generated journalistic text (380 characters)—*"Roma si è svegliata sotto un cielo grigio, ma l'entusiasmo dei tifosi non si è spento. 'Oggi vinciamo noi', mi ha detto Marco…"*—misclassified as human-written because it successfully mimics discourse markers such as direct quotations and emotional language.

These failures reveal that the model conflates *stylistic register* with authorship, failing when human writing is highly formal or when LLMs produce contextually rich, conversational text. The instruction-tuning formulation helped by preserving the LLM's pre-trained language understanding, enabling detection of subtle coherence issues without task-specific feature engineering. However, the 1,024-token truncation and single-pass inference prevent the model from leveraging document-level patterns, and the balanced training made the model overconfident on edge cases.

6.2. Subtask B

Table 8 presents the aggregated token-level confusion matrix across the validation set. With 94.1% token-level accuracy, the model reliably distinguishes human from machine tokens, but errors concentrate near the boundary. Stratifying character-level MAE by human-prefix length (Table 9) reveals a clear centering bias: the model over-predicts boundaries for short prefixes (positive bias up to +73 characters for 0–100 char prefixes) and under-predicts for long ones (negative bias of −53 characters for 400–512 char prefixes).

Analysis of high-error cases identifies three dominant patterns: smooth stylistic transitions (38%) where the LLM seamlessly continues the topic with minimal discontinuity, causing delayed detection; formal human writing with LLM-like features (27%) that triggers premature false positives; and noisy oscillating predictions near the boundary (21%) where the first-positive heuristic selects a premature transition. Two examples demonstrate these patterns. In an over-prediction case (+47 chars), the text *"…il governo ha annunciato nuove misure. [TRUE] Le riforme prevedono un incremento…[PRED]"* shows the LLM maintaining the same formal register, causing the model to delay detection until encountering

more stereotypical machine markers. In an under-prediction case (−32 chars), the text *"Il rapporto dell'ISTAT evidenzia che [PRED] i tassi di disoccupazione [TRUE] hanno raggiunto…"* shows human-authored statistical reporting triggering premature detection due to domain-specific terminology the model associates with LLM outputs.

Both errors stem from the binary token-labeling scheme's assumption of sharp stylistic discontinuity—an assumption violated when human and machine text share similar registers. Token-level supervision provided dense feedback that enabled fine-grained stylistic pattern learning, and UmBERTo's monolingual pre-training captured Italian-specific morphological cues (verb conjugations, article agreement) that signal authorship changes. However, the binary labeling assumes a sharp boundary that may not exist in practice, and the first-positive decoding commits to early predictions even under model uncertainty.

6.3. Cross-Task Insights and Takeaways

Comparing errors across both subtasks reveals a shared vulnerability: formal, templated human writing is systematically confused with machine-generated text. This suggests that current detection methods—including ours—rely heavily on stylistic fluency as a proxy for machine authorship, a heuristic that degrades as LLMs improve and as human writing becomes more formulaic.

Our analysis yields three main findings: (1) monolingual encoders outperform multilingual alternatives for Italian MGT detection, likely due to better morphological modeling; (2) token-level supervision is effective, but centering bias and smooth transitions remain challenging; and (3) truncation-based errors are non-negligible, as the first 512–1,024 tokens may not always contain decisive signals. Potential mitigations include length-aware calibration to reduce centering errors, multi-span inference for long documents, and soft boundary modeling to better reflect gradual human-machine transitions. These directions remain future work; our current system prioritizes simplicity and strong baseline performance.

7. Conclusion

This paper presented the *Stochastic Gradient Descenders* submission to the DeSegMa-IT shared task. We proposed a dual-strategy system: a parameter-efficient, prompt-based LLM (Qwen2.5) for binary detection and a dense token-classification approach using a monolingual Italian encoder (UmBERTo) for boundary localization. Our results—ranking **1st** in Subtask B and achieving competitive accuracy in Subtask A—demonstrate the effectiveness of task-specific architectural choices. Specifically, we showed that monolingual models remain a powerful baseline for fine-grained analysis of specific languages, outperforming larger multilingual counterparts in segmentation tasks. Future work will investigate multi-task learning frameworks to jointly address detection and localization, as well as calibration techniques to mitigate length-related biases in segmentation.

Acknowledgments

We thank the DeSegMa-IT organizers for releasing the dataset and maintaining the evaluation platform.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: assist with grammar and spelling checks, paraphrase and reword sentences for clarity, and support code debugging during experiment implementation. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. Puccetti, A. Pedrotti, A. Esuli, DeSegMa-IT at EVALITA 2026: Overview of the Detection and Segmentation of Machine Generated Text in Italian Task, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026, p. to appear.

[2] F. Cutugno, A. Miaschi, A. P. Aprosio, G. Rambelli, L. Siciliani, M. A. Stranisci, EVALITA 2026: Overview of the 9th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026, p. to appear.

[3] Qwen Team, Qwen2.5 Technical Report, arXiv preprint arXiv:2412.15115 (2024).

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, W. Chen, LoRA: Low-Rank Adaptation of Large Language Models, arXiv preprint arXiv:2106.09685 (2021).

[5] Musixmatch Research, UmBERTo: an Italian Language Model trained with Whole Word Masking, https://huggingface.co/Musixmatch/umberto-commoncrawl-cased-v1, 2020. Accessed: 2025-12-31.

[6] UnslothAI, Unsloth: Efficient Fine-tuning and Reinforcement Learning for LLMs, https://github.com/unslothai/unsloth, 2024. Accessed: 2025-12-31.

[7] I. Loshchilov, F. Hutter, Decoupled Weight Decay Regularization, in: International Conference on Learning Representations (ICLR), 2019, pp. 1–18.

[8] T. Dettmers, M. Lewis, S. Shleifer, L. Zettlemoyer, 8-bit Optimizers via Block-wise Quantization, arXiv preprint arXiv:2110.02861 (2021).

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, arXiv preprint arXiv:1907.11692 (2019).

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[11] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Transformers: State-of-the-Art Natural Language Processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.

[12] P. He, X. Liu, J. Gao, W. Chen, DeBERTa: Decoding-enhanced BERT with Disentangled Attention, arXiv preprint arXiv:2006.03654 (2020).

[13] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.

[14] DeSegMa-IT Organizers, DeSegMa-IT Leaderboard (Results), https://desegma.github.io/results/, 2025. Accessed: 2025-12-31.