Team DUAN at PAN 2026: Fine-Tuning Qwen3.5-2B for Reasoning Trajectory Source and Safety Detection

Notebook for the PAN Lab at CLEF 2026

Duan Xianbing[1], Han Zhongyuan[1,*], Liang Zhankeng[1] and Liu Chang[1]

[1]*Foshan University, Foshan, China*

Abstract

This paper describes our submissions to the PAN 2026 Reasoning Trajectory Detection task. The task considers two complementary problems: identifying whether a reasoning trajectory was produced by a human or by an AI system, and detecting unsafe reasoning at both trajectory and step level. For source detection, our final system fully fine-tunes a local Qwen3.5-2B model with a binary classification head and obtained a submitted test feedback score of 0.80. As an ablation, freezing the Qwen backbone and training only the classification head reduced the submitted test feedback score to 0.66. We also report the development history that led to this system: an encoder-based pipeline built on XLM-RoBERTa reached a local validation macro-F1 of 0.90 but only 0.57 on the test leaderboard, and recovering to 0.67 required calibrating the predicted class distribution rather than changing the model, which indicates a substantial domain shift between the released validation data and the test set. For safety detection, we use a Qwen3.5-2B model adapted with LoRA because full-backbone fine-tuning was not practical on the available hardware after expanding the data into step-level training examples. The LoRA safety system jointly predicts trace-level labels and step-level unsafe markers and obtained a submitted test feedback score of 0.65. These results indicate that full backbone adaptation is useful for source detection, that predicted-distribution calibration is an effective response to validation–test distribution mismatch, and that parameter-efficient adaptation provides a practical compromise for the larger safety detection setting.

Keywords

reasoning trajectory detection, AI-generated text detection, safety detection, Qwen, LoRA, threshold calibration, PAN

1. Introduction

Large language models increasingly produce explicit intermediate reasoning before their final answers. These traces are useful for inspection, but they also introduce new forensic and safety questions: a reasoning trajectory may be generated by a human or an AI system, and an apparently acceptable final answer may still be preceded by unsafe or misaligned intermediate reasoning. Both questions matter in practice. From a forensic perspective, mathematical solutions, homework, and technical write-ups are now routinely produced with LLM assistance, and detection systems that only consider final answers ignore the richer stylistic signal contained in the reasoning itself. From a safety perspective, recent work shows that harmful content can be embedded in intermediate reasoning steps even when the final answer appears benign, so trajectory-level moderation must look inside the trace rather than only at its conclusion [1].

The PAN 2026 Reasoning Trajectory Detection task addresses these questions through two subtasks [2]. Subtask 1 (source detection) asks whether a given solution to a math problem was written by a human or generated by an AI system. Subtask 2 (safety detection) asks for an overall safety label for a reasoning trace and, in addition, a binary safe/unsafe label for every individual reasoning step. The two subtasks stress different capabilities: the first is a stylistic attribution problem under domain shift, while the second is a hierarchical classification problem in which step-level and trace-level decisions interact.

CLEF 2026 Working Notes, 21–24 September 2026, Jena, Germany

*Corresponding author.

15007473346@163.com (D. Xianbing); hanzhongyuan@gmail.com (H. Zhongyuan); oooo1390329007@163.com (L. Zhankeng); lc965024004@gmail.com (L. Chang)

We participated in both subtasks. Our final systems are based on Qwen3.5-2B, a small open-weight decoder model from the Qwen series [3], used as a discriminative encoder with task-specific classification heads. For Subtask 1, we fully fine-tune the model and compare it with a frozen-backbone classification-head ablation. For Subtask 2, we use low-rank adaptation (LoRA) [4], which was selected as a practical compromise under GPU memory and runtime constraints caused by the larger step-level training set.

Beyond the final systems, this notebook documents the development path, including results that did not survive contact with the test leaderboard. Our first Subtask 1 pipeline, an XLM-RoBERTa classifier [5] with problem–solution input, reached a local validation macro-F1 of about 0.90 but scored only 0.57 on the leaderboard; most of the recoverable gap was closed not by changing the model but by calibrating the predicted human/AI ratio through decision-threshold sweeps. Two seemingly reasonable interventions, removing the problem context and up-weighting the human class, both degraded leaderboard performance. We report these negative results because they shaped our final design and because they illustrate a validation–test distribution mismatch that other participants likely faced as well.

The contributions of this notebook are as follows:

- We describe a full fine-tuning recipe for Qwen3.5-2B with a binary head for reasoning trajectory source detection, which obtained a submitted test feedback score of 0.80, together with a frozen-backbone ablation (0.66) that quantifies the value of backbone adaptation.
- We describe a LoRA-adapted multi-task Qwen3.5-2B system for safety detection that jointly predicts trace-level and step-level labels and obtained a submitted test feedback score of 0.65, improving over an mBERT multi-task baseline on both levels of the local validation data.
- We document the calibration-centered development history for Subtask 1, including a threshold-sweep analysis that raised the leaderboard score of an encoder-based system from 0.57 to 0.67 without retraining, and two negative results (solution-only input and human-class loss up-weighting).
- We report an LLM-as-judge error analysis of validation misclassifications, which suggests that general-purpose LLMs reliably recognize obvious AI style but struggle with well-formatted human solutions.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the task and data. Section 4 gives an overview of both systems. Sections 5 and 6 present the methods for the two subtasks, including the Subtask 1 development history. Section 7 reports experiments and analysis, Section 8 discusses lessons learned and limitations, and Section 9 concludes.

2. Related Work

2.1. Detection of AI-Generated Text

Detecting machine-generated text is commonly framed as supervised classification over pretrained encoders such as BERT [6], XLM-RoBERTa [5], and DeBERTa [7]. A recurring finding in this line of work is that in-domain accuracy is easy to obtain while generalization to unseen generators and domains is the actual bottleneck. FAID [8] addresses this with multi-task auxiliary learning over fine-grained generator labels and multi-level contrastive learning, and reports that generalizing to unseen domains remains substantially harder than generalizing to unseen generators. Our Subtask 1 experience is consistent with this picture: the released training and validation data are mathematical, the test set is broader, and the dominant practical problem was distribution shift rather than in-domain separability. Related shared tasks at PAN have studied generative authorship verification in previous years [9]; the PAN 2026 task extends this direction from generic text to explicit reasoning trajectories.

2.2. Safety of Reasoning Traces

Reasoning-specific safety analysis is a recent research direction. ReasoningShield [1] formalizes moderation over chain-of-thought traces, releases a query–trace moderation benchmark, and trains compact

detectors with stepwise risk analysis, showing that answer-oriented moderation tools underperform on risks hidden in intermediate steps. UnsafeChain [10] emphasizes hard cases: prompts that reliably elicit harmful completions carry most of the training signal for safety behavior, and correcting unsafe completions is more effective than filtering them out. STAIR [11] takes an alignment perspective and shows that introspective, step-by-step risk analysis improves safety over direct refusal training. The design of PAN 2026 Subtask 2, which labels every reasoning step in addition to the whole trace, reflects the same underlying observation: trajectory safety is a property of the step sequence, not only of the final answer. Our Subtask 2 system adopts the joint step-level and trace-level supervision suggested by this line of work, implemented as a multi-task classifier with a shared backbone.

3. Task and Data

3.1. Subtask 1: Source Detection

Subtask 1 provides a math problem and its solution. The system predicts whether the solution was written by a human or generated by an AI system. The released training split contains 87,556 instances, including 30,392 human examples and 57,164 AI examples. The validation split contains 497 examples, with 100 human examples and 397 AI examples from four model families (K2-Think V2, GPT-5 Nano, Gemini-3 Flash, and DeepSeek R1, following the organizers' data description). The test split contains 2,617 unlabeled examples. According to the task description, test queries may extend beyond mathematics to coding and real-life financial reasoning, whereas the released training and validation data are mathematical; this asymmetry turned out to be central to our development experience.

The official ranking metric is macro-F1 over the `human` and `ai` classes:

$$\text{Macro-}F_1 = \frac{F_1(\text{human}) + F_1(\text{ai})}{2}. \tag{1}$$

Because the metric averages the two class-wise F1 scores, a system that over-predicts the majority `ai` class is penalized heavily on the `human` class, which makes the predicted class distribution an important tuning target in addition to raw accuracy.

3.2. Subtask 2: Safety Detection

Subtask 2 provides a user query and a reasoning trace segmented into steps. Each trace has an overall label from `safe`, `potentially unsafe`, and `unsafe`, and each reasoning step has a binary safe/unsafe label. Notably, a trace can be labeled safe while containing individually unsafe steps, and vice versa, so the two levels of annotation are related but not reducible to each other. According to the task description, the queries cover three categories: risky queries that directly request harmful content, jailbreak-style queries in which the risk is obscured, and benign queries that merely contain risky tokens.

The training split contains 6,998 traces: 3,148 safe, 1,387 potentially unsafe, and 2,463 unsafe. The validation split contains 2,200 traces: 1,238 safe, 334 potentially unsafe, and 628 unsafe. The test split contains 2,835 traces. During training, the traces are expanded into step-level examples, which increases the effective number of training instances by roughly an order of magnitude and has direct consequences for the feasible adaptation strategy (Section 6). Table 1 summarizes the data statistics.

Two metrics are used by the organizers. The primary ranking metric is a macro-F1 over the overall trace label. The secondary metric is a per-sample soft F1 over the predicted step labels: for each sample with predicted step labels $\hat{y}_j$ and gold labels y_j,

$$F_1^{(i)} = \frac{2\,tp}{2\,tp + fp + fn}, \quad tp = \sum_j \hat{y}_j y_j, \quad fp = \sum_j \hat{y}_j (1 - y_j), \quad fn = \sum_j (1 - \hat{y}_j)\, y_j, \tag{2}$$

averaged over all samples. The step-level metric therefore rewards systems that localize unsafe content at the correct steps rather than only producing a correct overall verdict.

Table 1
Dataset statistics of the PAN 2026 Reasoning Trajectory Detection task as released to participants.

Subtask	Split	Instances	Class distribution	Unit
1	training	87,556	30,392 human / 57,164 ai	problem–solution pair
1	validation	497	100 human / 397 ai	problem–solution pair
1	test	2,617	unlabeled	problem–solution pair
2	training	6,998	3,148 safe / 1,387 pot. unsafe / 2,463 unsafe	trace
2	validation	2,200	1,238 safe / 334 pot. unsafe / 628 unsafe	trace
2	test	2,835	unlabeled	trace

4. System Overview

Our systems follow a discriminative fine-tuning pipeline, summarized in Figure 1. Each input is normalized by collapsing excessive whitespace and preserving the textual content of the solution or reasoning trace. Both final systems share the same backbone family and the same pooling scheme: the input is encoded by Qwen3.5-2B [3], the hidden state of the last non-padding token is taken as the sequence representation, and small linear heads on top of this representation produce the task logits. We train with cross entropy, using class weights where class imbalance is material, and we calibrate decision thresholds on the validation set for Subtask 1, because the operating point has a strong effect on human recall and macro-F1.

The two subtasks use the backbone in different adaptation modes. Subtask 1 uses full fine-tuning with a binary head; a frozen-backbone variant of the same architecture serves as an ablation. Subtask 2 keeps the backbone frozen and injects LoRA adapters [4], with two prediction heads for step-level and trace-level labels. This split was not a purely scientific choice: full fine-tuning was affordable for Subtask 1 but not for the step-expanded Subtask 2 training set on our hardware, a single NVIDIA RTX 3090 with 24 GB of memory. We view the resulting comparison as informative in itself, because it reflects the compute regime of many shared-task participants.

A second design principle emerged during development: submitted leaderboard feedback, not local validation, was treated as the final evidence for model selection. As Section 5.1 shows, local validation scores overstated test performance by 20 to 30 macro-F1 points for the encoder-based systems, and interventions that looked reasonable on validation data degraded test scores. All final decisions, including the choice of the full fine-tuning run as the primary Subtask 1 system, were made on submitted feedback.

5. Subtask 1: Source Detection

5.1. Development History: Encoder Baselines and Distribution Calibration

Our first pipeline followed the natural recipe for this task: an XLM-RoBERTa-based classifier [5], warm-started from the organizers' baseline model, with the problem and solution concatenated as `Problem: ... Solution: ...`, class-balanced loss, three epochs of training, and a maximum sequence length of 256. This model reached a local validation macro-F1 of about 0.9000. Its first leaderboard submission, however, scored only 0.57 macro-F1 with 0.66 accuracy.

Inspection of the submitted predictions showed a strongly skewed distribution: only 12.6% of test instances were predicted as human, although the validation split contains about 20% human instances and the validation confusion matrix already showed weak human recall (77 of 100 human examples correct, versus 390 of 397 AI examples). Because macro-F1 weighs the human class equally, an over-prediction of `ai` of this magnitude is costly. We therefore swept the decision threshold on the AI probability upward and generated multiple submissions whose predicted human ratios covered the plausible range. Figure 2 shows the resulting relationship between the predicted human ratio and the leaderboard score: the score rises from 0.57 at the default operating point to 0.67 when about 26% of

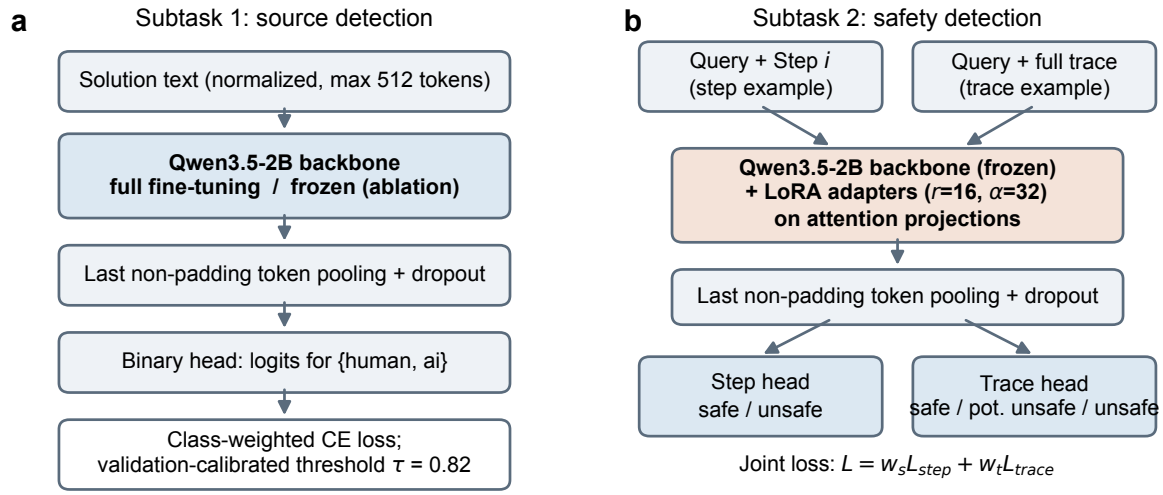

Figure 1: System architectures. (a) Subtask 1: the solution text is encoded by Qwen3.5-2B, pooled at the last non-padding token, and classified by a binary head; the backbone is either fully fine-tuned (final system) or frozen (ablation). The decision threshold is calibrated on the validation split. (b) Subtask 2: a shared Qwen3.5-2B backbone with frozen weights and LoRA adapters encodes step examples (query plus a single step) and trace examples (query plus the full trace); a binary step head and a three-way trace head are trained jointly with a weighted cross-entropy objective.

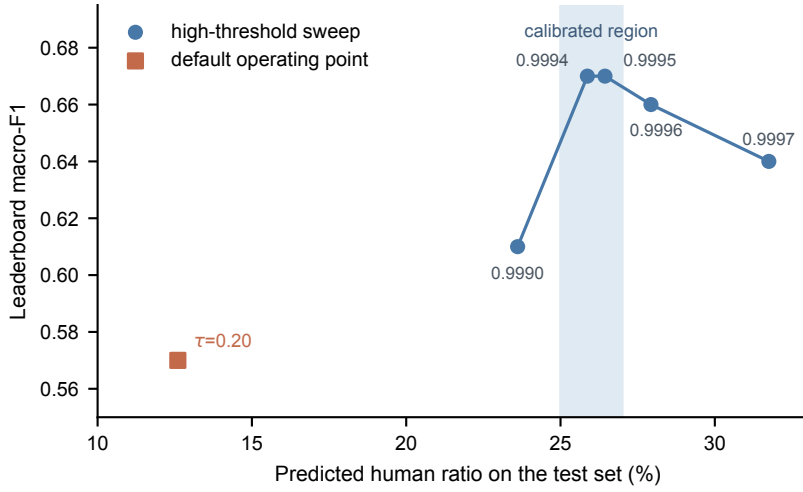

Figure 2: Distribution calibration for the encoder-based Subtask 1 system. Each point is a leaderboard submission of the same XLM-RoBERTa model under a different decision threshold on the AI probability (labels next to the points). Moving the predicted human ratio from 12.6% to about 26% raised the leaderboard macro-F1 from 0.57 to 0.67 without retraining; pushing further reversed the gain.

test instances are predicted as human, and declines again beyond 28%. The required thresholds were extreme (0.9990 to 0.9997 on the AI probability), which indicates that the model's probabilities were poorly calibrated for the test domain even though its ranking of instances retained useful signal.

Two further interventions, both plausible a priori, failed on the leaderboard. First, we trained a solution-only variant to reduce the influence of domain-specific problem statements; it reached a validation macro-F1 of 0.8955 but only about 0.61 on the leaderboard, worse than the calibrated problem–solution model, which suggests that the problem context still contributes useful signal. Second, we increased the loss weight of the human class by a factor of 1.2 to attack the weak human recall directly; validation macro-F1 dropped to 0.8789 with a human recall of about 0.73, and the leaderboard score fell to 0.57. We concluded that reshaping the decision boundary during training was more damaging

than adjusting the operating point after training, and we did not pursue larger multipliers.

We also ran an LLM-as-judge error analysis on 30 misclassified validation examples, asking two DeepSeek chat models to re-decide each case. Both models corrected essentially all cases in which an AI-generated solution had been misclassified as human, but corrected at most 17% of the cases in which a human solution had been misclassified as AI (36.67% overall correction rate for the stronger of the two). The pattern of the residual errors indicates that well-structured human solutions with regular formatting and LaTeX notation are systematically mistaken for AI output, by our classifier and by the judging LLMs alike. Based on this asymmetry, and to avoid the compliance risk of routing test data through a closed-source API, we used the LLM judges only for error analysis and not in any submitted pipeline.

The overall lesson of this phase was that the binding constraint was not in-domain separability but transfer: the leaderboard rewarded systems whose predicted distribution matched the test set, and the cheapest reliable improvement came from calibration rather than from architecture or loss changes. This motivated the move to a stronger backbone with full adaptation, described next, while keeping threshold calibration in the pipeline.

5.2. Final System: Full Fine-Tuning of Qwen3.5-2B

For the final Subtask 1 system, we use the solution text only as input. Earlier experiments also considered concatenating the problem and solution; this helped the XLM-RoBERTa pipeline, but it did not generalize as well in our later Qwen and DeBERTa [7] runs, so the final configuration drops the problem context. Each instance is converted into a binary label: `human` is mapped to 0 and `ai` to 1. Input texts are normalized by collapsing repeated whitespace and blank lines.

The model attaches a binary linear classification head to the last non-padding token representation of Qwen3.5-2B (Figure 1a). We fine-tune all parameters for three epochs with a maximum sequence length of 512, batch size 1 with gradient accumulation over 16 steps (effective batch size 16), learning rate 5×10^{-6} with a cosine schedule, and class weights computed from the training split to reduce the effect of the human/AI imbalance. To fit full fine-tuning of a 2.21-billion-parameter model into 24 GB of GPU memory, we enable fp16 mixed precision, gradient checkpointing, and the memory-efficient Adafactor optimizer. After training, the decision threshold on the AI probability is calibrated on the validation split; the selected threshold is 0.82, which is far less extreme than the thresholds required by the encoder-based pipeline and suggests better-calibrated probabilities. Table 2 summarizes the configuration of both final systems.

5.3. Frozen-Backbone Ablation

We also train a frozen-backbone ablation in which all Qwen backbone parameters are fixed and only the final binary classification head is trainable. This setting updates 4,098 parameters out of roughly 2.21 billion, is much cheaper than full fine-tuning, and serves as a direct ablation of whether Qwen's frozen representation is sufficient for the source-detection task. A verification script asserts that no backbone parameter receives gradients, so the comparison isolates the effect of backbone adaptation.

6. Subtask 2: Safety Detection

6.1. Multi-Task Architecture

For Subtask 2, we train a multi-task classifier with a shared Qwen3.5-2B backbone and two prediction heads (Figure 1b). The step head predicts binary `safe step` versus `unsafe step`; the trace head predicts `safe`, `potentially unsafe`, or `unsafe`. Reasoning traces are segmented into steps by matching the explicit `Step` n`:` markers used in the data. Each step example encodes the user query together with the single step (`Query: ... Step` i`: ...`), so that the safety of a step is always judged in the context of the query it responds to; this matters because many steps are only unsafe

Table 2
Configuration of the two final systems. Both share the Qwen3.5-2B backbone and last non-padding token pooling.

Setting	Subtask 1 (full fine-tuning)	Subtask 2 (LoRA)
Trainable parameters	all (≈2.21B)	LoRA adapters + two heads
LoRA rank / alpha / dropout	–	16 / 32 / 0.05
LoRA target modules	–	attention projections
Prediction heads	binary (human/ai)	step (2-way), trace (3-way)
Max sequence length	512	256
Epochs	3	2
Batch size × grad. accumulation	1×16	1×8
Learning rate	5×10^{-6}	2×10^{-4}
LR schedule	cosine	cosine
Class weighting	from training distribution	balanced step sampling
Decision threshold	0.82 (validation-calibrated)	argmax
Seed	42	42

insofar as they advance a harmful intent expressed in the query. Trace examples encode the query together with the full reasoning trace. Both example types share the backbone and pooling; a task indicator routes each example to its head, and the training loss is a weighted combination of the step-level and trace-level cross-entropy terms.

Step expansion creates a much larger and imbalanced training set: safe steps far outnumber unsafe ones. We therefore cap the number of step examples relative to the number of trace examples and balance the retained step examples between the safe and unsafe classes by sampling, which keeps the two tasks at comparable scale during joint training.

6.2. Adaptation Strategy: Why LoRA

Full-backbone fine-tuning was not practical for this subtask under our available hardware. The step-level expansion multiplies the effective number of training examples well beyond the trace-level data alone, and the memory and runtime requirements of full Qwen3.5-2B fine-tuning were too high for stable iteration on a single RTX 3090. Therefore, our final system keeps the backbone frozen and trains LoRA adapters [4] on the attention projection matrices, with rank 16, scaling factor $\alpha = 32$, and dropout 0.05, together with the two classification heads. The model is trained for two epochs with maximum sequence length 256, batch size 1 with gradient accumulation over 8 steps, and learning rate 2×10^{-4}.

6.3. Inference

At test time, each trace is segmented with the same step-splitting procedure as in training. The trace head produces the overall label by argmax over the three classes, and the step head produces one binary label per step, which are concatenated into the detailed label sequence required by the submission format. No threshold calibration is applied in this subtask, because the released feedback does not expose the class-wise behavior needed to guide it.

7. Experiments and Results

7.1. Experimental Setup

All models were trained on a single NVIDIA RTX 3090 (24 GB) with fixed random seed 42, using PyTorch and Hugging Face Transformers, plus PEFT for the LoRA runs. For Subtask 1 we report macro-F1;

Table 3
Local validation results. The XLM-RoBERTa development systems are included for completeness of the Subtask 1 record; the mBERT system is the organizers-style multi-task baseline we trained for Subtask 2.

Subtask	System	Main setting	Metric	Score
1	XLM-R (development)	problem + solution	macro-F1	0.9000
1	XLM-R (development)	solution only	macro-F1	0.8955
1	XLM-R (development)	human loss weight 1.2	macro-F1	0.8789
1	Qwen3.5-2B frozen head	backbone frozen, head only	macro-F1	0.8139
1	Qwen3.5-2B full fine-tuning	solution only, threshold 0.82	macro-F1	0.9297
2	mBERT multi-task baseline	step + trace heads	step macro-F1	0.6719
2	mBERT multi-task baseline	step + trace heads	trace macro-F1	0.6142
2	Qwen3.5-2B LoRA	rank 16 adapters	step macro-F1	0.7683
2	Qwen3.5-2B LoRA	rank 16 adapters	trace macro-F1	0.6708

Table 4
Submitted test results. For the XLM-RoBERTa development systems, the best score across the swept decision thresholds is shown.

Subtask	System	Submitted test feedback score
1	XLM-R, problem + solution (calibrated)	0.67
1	XLM-R, solution only	0.61
1	XLM-R, human loss weight 1.2	0.57
1	Qwen3.5-2B frozen head	0.66
1	Qwen3.5-2B full fine-tuning	0.80
2	Qwen3.5-2B LoRA	0.65

for Subtask 2 we report macro-F1 at both step and trace level on the validation data, and the official feedback score for the test set. The submitted test feedback scores are returned by the evaluation platform at two-decimal precision. Unless noted otherwise, every reported configuration is a single training run; we did not have the compute budget for multi-seed replication, so small differences should be read with caution.

7.2. Subtask 1 Results

Table 3 reports local validation metrics and Table 4 the submitted test feedback. The frozen Qwen ablation performs substantially below full fine-tuning on validation data (0.8139 versus 0.9297 macro-F1), and the gap persists on the test set (0.66 versus 0.80). Notably, the frozen 4,098-parameter head on top of an unadapted Qwen representation already matches the calibrated test score of the fully trained XLM-RoBERTa pipeline (0.66 versus 0.67), which suggests that the pretrained Qwen representation encodes much of the relevant stylistic signal; however, closing the remaining 14-point gap required updating the backbone.

Figure 3 places all five Subtask 1 systems from our development history side by side. Two patterns stand out. First, all systems lose macro-F1 between validation and test, consistent with the domain shift discussed in Section 5.1, but the size of the loss varies: the encoder-based systems lose 23 to 31 points, while the fully fine-tuned Qwen system loses about 13. Second, the ranking of systems by validation score does not match their ranking by test score; the solution-only XLM-RoBERTa model, for example, validates within half a point of the problem–solution model but tests 6 points lower. Local validation alone was therefore not sufficient for model selection, and we treated the submitted feedback as the final evidence for selecting the Qwen full fine-tuning run as the main Subtask 1 system.

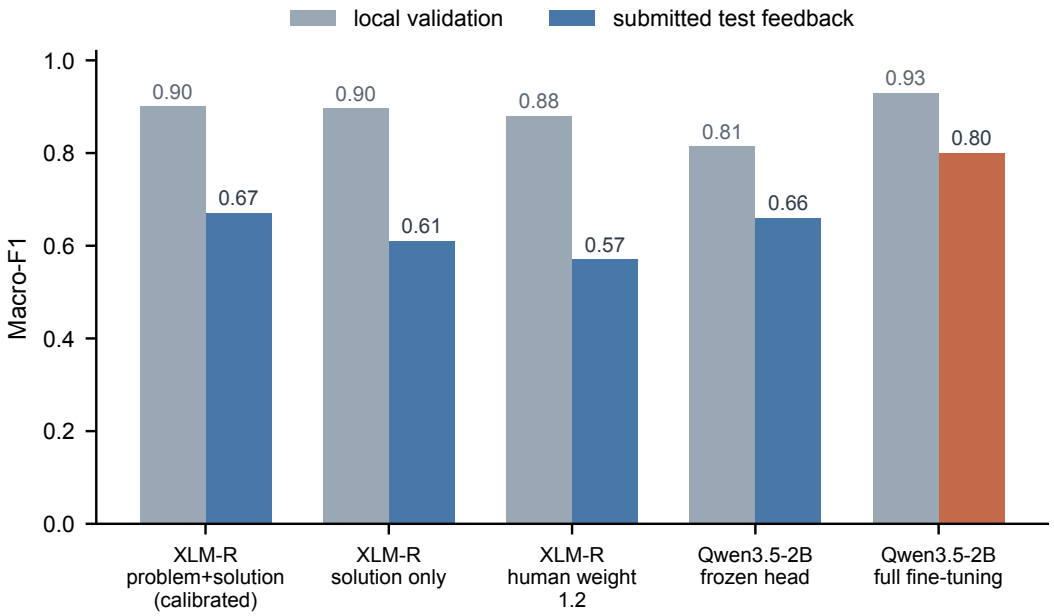

Figure 3: Local validation versus submitted test macro-F1 for all Subtask 1 systems in our development history. Every system loses performance under the validation–test domain shift, but full fine-tuning of Qwen3.5-2B (right) both scores highest and shows the smallest gap. Validation ranking does not predict test ranking for the encoder-based systems.

7.3. Subtask 2 Results

For Subtask 2, Qwen with LoRA improves over the mBERT [6] multi-task baseline on both levels of the validation data: step macro-F1 rises from 0.6719 to 0.7683 and trace macro-F1 from 0.6142 to 0.6708 (Figure 4). The improvement is larger at step level than at trace level, which is plausible given the architecture: each step example is short and query-conditioned, so a stronger backbone representation translates directly into better step decisions, whereas the trace decision must compress a long, heterogeneous input through a single pooled vector. The final system obtained a submitted test feedback score of 0.65.

The per-class validation behavior shows the expected difficulty ordering: clearly safe and clearly unsafe traces are separated reasonably well, while the `potentially unsafe` class, which sits between them by definition, remains the weakest. Improving this boundary class is, in our view, the highest-leverage direction for future work on this subtask.

7.4. Error Analysis

For Subtask 1, the validation confusion of the development-stage system (77/100 human correct, 390/397 AI correct) and the LLM-as-judge analysis of Section 5.1 point at the same hard region: human-written solutions that are complete, well structured, and typeset with LaTeX are misclassified as AI, both by our classifiers and by general-purpose LLM judges. This is an inherent difficulty of the task rather than an artifact of one model: as human writing conventions in mathematics converge on the same clean, step-by-step format that LLMs produce, surface style becomes less informative, and the residual signal is subtle. The asymmetry also explains why post-hoc calibration was so effective: the models rank instances reasonably well but systematically shift probability mass toward the AI class for well-formatted text.

For Subtask 2, the main difficulty is the combination of trace-level and step-level supervision under label interaction: some traces are safe overall despite containing unsafe steps, and vice versa, so the trace head cannot be replaced by a simple aggregation of step predictions. Our system trains the two

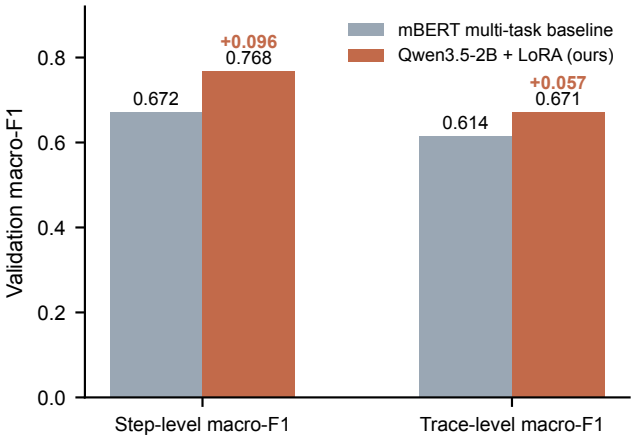

Figure 4: Subtask 2 validation results. The LoRA-adapted Qwen3.5-2B multi-task system improves over the mBERT multi-task baseline at both step level and trace level, with the larger gain at step level.

heads jointly but does not model the aggregation explicitly, and the trace-level results suggest that a hierarchical aggregation mechanism over step representations, or consistency constraints between the two levels as proposed in reasoning-trace moderation work [1], could recover part of the remaining gap.

8. Discussion

The Subtask 1 results support three practical conclusions. First, full fine-tuning matters for reasoning trajectory source detection: although the frozen Qwen representation is already competitive with a fully trained encoder pipeline, training only the final head leaves a 14-point gap in submitted test feedback. Updating the backbone appears to help the model adapt to the specific stylistic and structural signals that distinguish human-written solutions from AI-generated solutions.

Second, under validation–test domain shift, the predicted class distribution is a first-order concern for macro-F1. The cheapest reliable improvement of the entire development cycle, from 0.57 to 0.67, came from sweeping a decision threshold, and the systematic failure of training-time interventions (solution-only input, human-class up-weighting) against this simple post-hoc calibration was the most instructive negative result of our participation. We would recommend distribution inspection and threshold sweeps as the first diagnostic step for any participant whose validation and test scores diverge.

Third, local validation alone was not sufficient for model selection; several systems with strong validation results had weaker submitted test scores, and the validation ranking did not predict the test ranking. Treating leaderboard feedback as the primary selection evidence is a pragmatic response, but it consumes limited submission quota and risks a mild form of test-set overfitting; with more submissions we would have preferred to hold some feedback in reserve for final confirmation.

For Subtask 2, LoRA allowed us to adapt Qwen3.5-2B within the available compute budget after step-level expansion made full-backbone training impractical, and it outperformed the mBERT multi-task baseline at both granularities. The gap between the step-level and trace-level improvements, together with the persistent weakness of the `potentially unsafe` class, suggests that stronger hierarchical aggregation and more targeted treatment of boundary cases could further improve the system, in line with the hard-case emphasis of recent safety training work [10, 11].

Several limitations qualify these conclusions. All results are single runs with one seed, so we cannot separate configuration effects from run-to-run variance. The submitted test feedback is a two-decimal aggregate, which limits the resolution of test-side comparisons. The full fine-tuning versus LoRA comparison is confounded across subtasks: we did not run full fine-tuning on Subtask 2 or LoRA on Subtask

1, so our results show that each strategy was adequate in its setting, not that it was superior to the alternative there. Finally, our error analysis of the human/AI boundary is based on a small sample of 30 validation errors.

9. Conclusion

We presented two Qwen-based systems for PAN 2026 Reasoning Trajectory Detection, together with the development history that produced them. For source detection, full fine-tuning of Qwen3.5-2B achieved our best submitted test feedback score of 0.80 and clearly outperformed a frozen-backbone classification-head ablation (0.66); the preceding encoder-based development phase showed that predicted-distribution calibration under domain shift can be worth ten macro-F1 points on the leaderboard without any retraining. For safety detection, a LoRA-adapted Qwen3.5-2B multi-task classifier achieved a submitted test feedback score of 0.65 while remaining feasible on a single consumer GPU, and improved over an mBERT multi-task baseline at both step and trace level.

The results support two practical conclusions: full backbone adaptation is valuable for source detection, and parameter-efficient adaptation is a useful compromise for larger step-level safety detection pipelines. For future editions of the task, we plan to investigate multi-seed ensembles for variance reduction, stylometric features as a complement to neural representations for the well-formatted-human hard cases, and hierarchical step-to-trace aggregation with explicit consistency modeling for safety detection.

Acknowledgments

We thank the PAN 2026 organizers for preparing the task, datasets, and evaluation platform. This work is supported by the National Social Science Foundation of China (24BYY080).

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5.5 to assist with code implementation, manuscript drafting, language polishing, and LaTeX editing. The authors reviewed, revised, and verified the AI-assisted outputs, retained full responsibility for the study design, experiments, analysis, code, and manuscript, and take full responsibility for the publication's content.

References

[1] C. Li, J. Wang, X. Pan, G. Hong, M. Yang, ReasoningShield: Safety detection over reasoning traces of large reasoning models, arXiv preprint arXiv:2505.17244 (2025). doi:10.48550/arXiv.2505.17244.

[2] PAN 2026 Organizers, PAN 2026 Reasoning Trajectory Detection, https://pan.webis.de/clef26/pan26-web/reasoning-trajectory-detection.html, 2026. Accessed: 2026-05-24.

[3] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al., Qwen3 Technical Report, arXiv preprint arXiv:2505.09388 (2025).

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[5] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8440–8451. doi:10.18653/v1/2020.acl-main.747.

[6] 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. doi:10.18653/v1/N19-1423.

[7] P. He, X. Liu, J. Gao, W. Chen, DeBERTa: Decoding-enhanced BERT with disentangled attention, arXiv preprint arXiv:2006.03654 (2020). doi:10.48550/arXiv.2006.03654.

[8] M. N. Ta, D. C. Van, D.-A. Hoang, M. Le-Anh, T. Nguyen, M. A. Tran Nguyen, Y. Wang, P. Nakov, D. V. Sang, FAID: Fine-grained AI-generated text detection using multi-task auxiliary and multi-level contrastive learning, in: Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), 2026, pp. 3275–3296. doi:10.48550/arXiv.2505.14271.

[9] J. Bevendorff, D. Dementieva, M. Fröbe, B. Gipp, A. Greiner-Petter, J. Karlgren, M. Mayerl, P. Nakov, A. Panchenko, M. Potthast, A. Shelmanov, E. Stamatatos, B. Stein, Y. Wang, M. Wiegmann, E. Zangerle, Overview of PAN 2025: Generative AI Authorship Verification, Multi-Author Writing Style Analysis, Multilingual Text Detoxification, and Generative Plagiarism Detection, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2025), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2025.

[10] R. V. Tomar, P. Nakov, Y. Wang, UnsafeChain: Enhancing reasoning model safety via hard cases, arXiv preprint arXiv:2507.21652 (2025). doi:10.48550/arXiv.2507.21652.

[11] Y. Zhang, S. Zhang, Y. Huang, Z. Xia, Z. Fang, X. Yang, R. Duan, D. Yan, Y. Dong, J. Zhu, STAIR: Improving safety alignment with introspective reasoning, in: Proceedings of the 42nd International Conference on Machine Learning, 2025. doi:10.48550/arXiv.2502.02384.