An Agentic Multi-LLM Intersection Approach to Historical Spanish Emotion Detection

Francisco-Javier Rodrigo-Ginés[1,*], Jorge Chamorro-Padial[2]

[1]NLP & IR, Universidad Nacional de Educación a Distancia (UNED), 28040 Madrid, Spain
[2]OpenCódice Research, Spain

Abstract

We describe the submission to HISEMOTIONS 2026 at IberLEF, a multi-label emotion-detection task on sixteenth- and seventeenth-century Spanish epistolary prose. The entire experimentation pipeline was driven by an *agentic* research loop in which a coding agent built on Claude Opus 4.7 received the task description, the raw data and API credentials, and iterated through a sequence of strategies under researcher supervision. The data has a structural irregularity that dominated the campaign: the 2660-instance training partition is LLM-annotated and the 425-instance development and 863-instance test partitions are human-annotated, with per-emotion frequencies that differ by an order of magnitude in some classes. The agent characterised this distribution shift in the first iteration and elected to use prompted large language models throughout, holding fine-tuned encoders as a fall-back that never proved competitive. To close the gap between the small human-annotated development set and the test set, the agent compared three strategies: a per-class dev-tuned ensemble of three LLMs, a targeted re-labelling of putative false negatives by a second model, and a meta-prompted variant with explicit hard-case demonstrations. All three either overfitted the development set or returned a negative result. The strategy that produced the highest test score was, with hindsight, the simplest one we considered: the pure label-wise intersection of the binary outputs of GPT-5.4 and GPT-5.5. The intersection scored 0.5775 macro-F1 on the public test set. We present the result as a generalisable finding about multi-label emotion classification with limited human-annotated development data: when a development set is small, per-class tuning is a worse default than the parameterless intersection of two strong, diverse classifiers.

Keywords

Multi-label emotion classification, Historical Spanish, Agentic research, Multi-LLM intersection, Distribution shift, IberLEF 2026

1. Introduction

HISEMOTIONS 2026 [1] asks participating systems to assign a subset of seven emotion labels to short text fragments drawn from sixteenth- and seventeenth-century Spanish private correspondence. The label inventory comprises six emotions in the Ekman tradition (anger, fear, joy, sadness, surprise, hope) and a residual neutral or unclear class for fragments that carry no detectable emotional content. The task is part of the IberLEF 2026 shared-task forum [2] and the official metric is macro-F1 averaged across the six emotion labels, computed independently per label so that rare classes contribute as much to the score as frequent ones.

The submission was produced under a methodological choice that frames the rest of the paper. Instead of a single researcher iterating manually across model families, our entire experimentation loop was operated by a Claude Opus 4.7 coding agent under researcher supervision. The agent received the task description, the raw data, GPU access and API credentials, and was responsible for the full life cycle of every experiment: writing training scripts, launching them, parsing their logs, building submission files and updating its mental model in light of leaderboard feedback. The researcher held two non-negotiable veto points, namely the actual upload of a submission and any significant change of strategic direction. We refer to the resulting working mode as a *researcher-in-the-loop* variant of agentic research, and we describe it in detail in Section 4.

IberLEF 2026, September 2026, León, Spain
*Corresponding author.
frodrigo@invi.uned.es (Francisco-Javier Rodrigo-Ginés); jorge@opencodice.org (Jorge Chamorro-Padial)
0000-0001-6235-6860 (Francisco-Javier Rodrigo-Ginés); 0000-0002-6334-3786 (Jorge Chamorro-Padial)

The hypothesis that drives the work is stated plainly, because it is itself one of our contributions. We hypothesise that a coding agent, operated under a researcher-in-the-loop protocol, amplifies what a single researcher can explore, more experiments, faster, at a far lower mechanical cost, without displacing the two functions we hold to be non-delegable: the generation of hypotheses, that is, deciding which research direction is worth pursuing, and the acceptance criterion, that is, deciding what counts as a valid and publishable result. The HISEMOTIONS campaign is a test of that hypothesis on one shared task, and the division of labour through which we tested it is set out in Section 4.2.

The specific framing matters because HISEMOTIONS, despite being a narrow task in scope, exposes a feature of agentic research that is much more general than the historical-Spanish domain. The training partition is silently LLM-annotated, with systematic biases relative to the human convention used in the development and test partitions. A naive supervised system inherits those biases and pays for them in the metric. The first iteration of our loop was, by design, an exploratory data analysis that surfaced this irregularity, and the rest of the campaign was structured around it. We give the empirical record of the shift in Section 3 and the strategies that responded to it in Section 6.

The empirical contribution of the paper is independent of the methodological one. To close the dev-test gap, the agent compared three substantive strategies and submitted the strongest. The strongest, with hindsight, was also the simplest: the label-wise intersection of two strong prompted models (GPT-5.4 and GPT-5.5) with no per-class tuning. The intersection scored 0.5775 on the official test set, ahead of strategies whose dev macro-F1 was higher but whose test scores were lower because the strategies had overfitted the small development partition. We treat the comparison between the intersection and the per-class tuned ensemble as the most useful finding to communicate to future participants of comparable tasks.

We organise the paper as follows. Section 2 positions the work in the agentic-research literature, in the multi-label emotion-classification literature, and in the distribution-shift literature that frames the central data-level challenge. Section 3 describes the task in formal terms and presents an exploratory analysis of the three splits. Section 4 is the central methodological contribution: the researcher-in-the-loop protocol, the autonomy spectrum it inhabits, the operational details, and a catalogue of the failure modes we observed. Section 5 describes the submitted system in detail. Section 6 reports every strategy the agent explored, the results that came back from the leaderboard, and the empirical observation that motivated this paper. Finally, Section 7 reads the campaign as a sequence of transferable lessons for future agentic submissions, and Section 9 concludes.

2. Related Work

The work belongs to the intersection of three lines of research: agentic research with language models, multi-label emotion classification with limited supervision, and dataset-shift theory applied to text classification. We sketch each in turn, with emphasis on the empirical pattern that connects them in our campaign.

The first line of work is the recent push to operate language models as autonomous research agents. Voyager [3] demonstrated that an LLM agent can drive an open-ended exploration of a complex environment without an externally specified curriculum. The AgentBench benchmarks [4] formalised LLM-as-agent evaluation across a wide variety of tasks, including database and operating-system interaction. Anthropic's recent computer-use interface [5] extends that capability to the desktop, with the human in an auditing role rather than as the operator of every individual command. Karpathy's *autoresearch* [6] sketches a research loop in which the agent autonomously proposes a research direction, writes the code, runs experiments and reports results, treating the human researcher as an inspector of conclusions rather than as a per-step operator. Our methodology accepts the same premise about per-experiment autonomy, but pushes back against full automation of the loop on the basis of evidence we present in Section 4.6: agents inherit failure modes from their underlying language model, and a researcher operating with strictly different priors is the cheapest available defence against them. A more general survey of language-model capabilities relevant to agentic operation appears in [7].

The second line of work is multi-label emotion classification. The discipline has been organised, in the last decade, around shared tasks at SemEval and IberLEF. SemEval-2018 Task 1 [8] established the multi-label formulation in social-media text and the EmoEvent corpus [9] extended it to multilingual event-anchored data. The most recent reference point is SemEval-2025 Task 11 [10] and its accompanying BRIGHTER dataset [11], which covers twenty-eight languages including Spanish. The strongest systems in that campaign were dominated by ensembles of large language models with various combination rules, including weighted voting and stacked meta-classifiers. The empirical observation we make in Section 6, namely that the parameterless intersection of two strong LLMs outperforms a per-class tuned ensemble of three on a held-out test set, is in methodological tension with that body of work and we believe it is most usefully interpreted as a development-set-size effect rather than as a general claim about ensembling.

The third line of work is the literature on dataset shift, and specifically the form of shift that arises when a training and evaluation partition are produced by different annotators. Quiñonero-Candela et al. [12] and the unifying treatment of Moreno-Torres et al. [13] provide the formal vocabulary; Riedel et al. [14] document a closely related phenomenon under the name *label noise from distant supervision*, in which the label distribution of a weakly-supervised training set diverges systematically from the human-annotated evaluation set. HISEMOTIONS sits inside this literature in a precise and somewhat unusual sense: the training partition is LLM-annotated rather than weakly supervised in the classical sense, and the resulting prior shift is large enough to make supervised fine-tuning on the training set actively harmful for performance on the human-annotated evaluation set. Plumbing-level details of multi-label classification with imbalanced classes, including the F1-thresholding theory of Lipton et al. [15] and the multi-label algorithmic survey of Zhang and Zhou [16], inform the design choices that we describe in Section 5.

The narrower line of historical-text emotion analysis remains relatively under-explored. Schmidt et al. [17] report a corpus of historical German plays with emotion annotations and discuss the cross-temporal generalisation problem that arises when modern emotion lexicons are applied to early modern text. Sprugnoli et al. [18] report a similar phenomenon for Italian and observe that lexicon methods underperform on early modern register. Our setting reproduces the observation: the historical Spanish of the HISEMOTIONS corpus is close enough to modern Spanish for contemporary LLMs to read it confidently, but far enough that domain-adapted modern Spanish encoders behave erratically when fine-tuned on the LLM-annotated training set.

3. Task, Data, and Exploratory Analysis

This section describes the task in formal terms, presents the splits and basic statistics of the dataset, and walks through an exploratory analysis whose central finding (the train versus human-annotated distribution mismatch) determined the shape of the rest of the campaign. The exploratory analysis is also the empirical record against which we will, in Section 6, compare the predictions of every strategy the agent tried.

3.1. Task definition and splits

Each instance of HISEMOTIONS 2026 consists of a short text fragment x drawn from a sixteenth- or seventeenth-century Spanish letter, and a binary vector $\mathbf{y} \in \{0, 1\}^6$ recording whether each of the six emotions (anger, fear, joy, sadness, surprise, hope) is present in the fragment. The "Nuetral/unclear" class is treated implicitly: a fragment with all-zero label vector is neutral by definition. A submission must produce, for each test instance, a binary prediction $\hat{\mathbf{y}}_t \in \{0, 1\}^6$. The official metric is the macro-averaged F_1 across the six emotion labels, which weights each label equally regardless of support and is therefore sensitive to the rare-class predictions in a way that sample-weighted metrics are not.

The dataset is partitioned into three splits: a training set of 2 660 instances annotated automatically by an LLM (the organisers acknowledge this in the dataset card), a development set of 425 instances annotated by humans, and a held-out test set of 863 instances also annotated by humans. The training

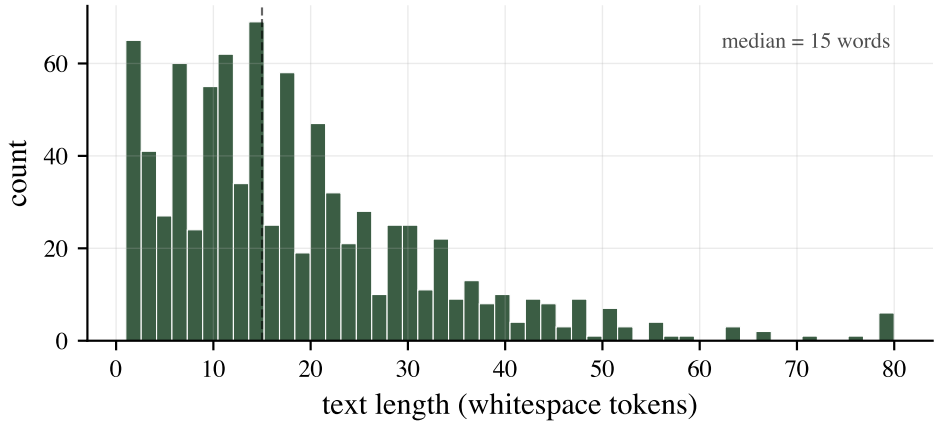

Figure 1: Distribution of text-fragment length on the test set, measured in whitespace tokens and clipped at 80 for readability. The dashed line marks the median (15 tokens). Most fragments fit comfortably inside any modern transformer context window, so encoder choice is not constrained by sequence length.

set is roughly six times larger than the development set, but, as we will see, this size advantage does not translate into a corresponding gain in usefulness, because the training labels are drawn from a different conditional distribution than the human-annotated splits.

3.2. Lengths and composition

The text fragments are short. Across the test set the median length is 15 whitespace tokens and the 95th percentile is 45 tokens; the longest fragment in the test set runs to 139 tokens but is exceptional. Figure 1 reports the empirical length distribution. The brevity has practical consequences for the system designer: a transformer encoder operating on these inputs has very little context to work with, and the linguistic register of a short early modern Spanish letter fragment is recognisably different from any of the corpora on which a contemporary Spanish encoder was pre-trained. Pretrained models that excel on contemporary Spanish often misread courtesy formulas (“beso las manos de Vuestra Merced”, literally “I kiss the hands of Your Grace”) as expressions of affection or joy, when these are conventional rather than emotionally loaded.

3.3. Class balance on the human-annotated splits

The label-frequency profile of the development and test splits is the second relevant piece of context. Figure 2 reports the absolute count of positive instances per emotion on each human-annotated split. The human convention is dominated by sadness and hope on dev, with sadness still dominant but neutral fragments being far more common on test. The two genuinely rare classes are anger and surprise; surprise has only seven positive instances on dev and two on test, which is small enough that ordinary statistical claims about that label are noise-dominated. The macro-F1 metric is, by construction, sensitive to all six labels equally, which means that a single misprediction on surprise can move the score by an outsized amount.

3.4. The train versus human-annotated distribution shift

The single most important property of the dataset is the mismatch between the LLM-annotated training partition and the two human-annotated partitions. Figure 3 reports per-emotion positive rates on each split. The mismatch is large and systematic. The training set predicts sadness on roughly forty-five per cent of fragments, against seventeen per cent on dev and nineteen per cent on test. The training set predicts anger on less than half of one per cent of fragments, against twelve per cent on dev. The training set predicts hope on under three per cent of fragments, against twenty per cent on dev. The qualitative pattern is that the LLM annotator that produced the training set systematically over-predicts sadness and

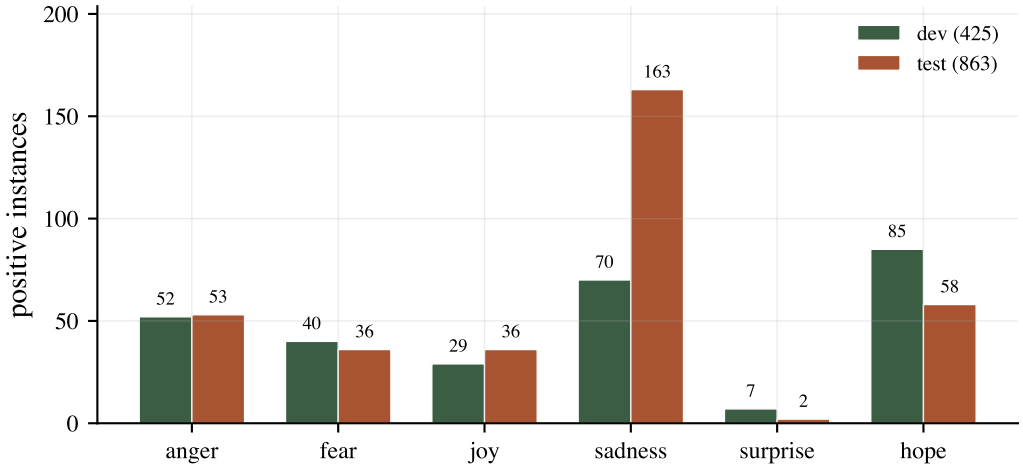

Figure 2: Number of positive instances per emotion on the human-annotated dev (425 fragments) and test (863 fragments) splits. Surprise is extremely rare on both splits, which makes macro-F1 numerically volatile in that label. Sadness is the most frequent emotion on both splits.

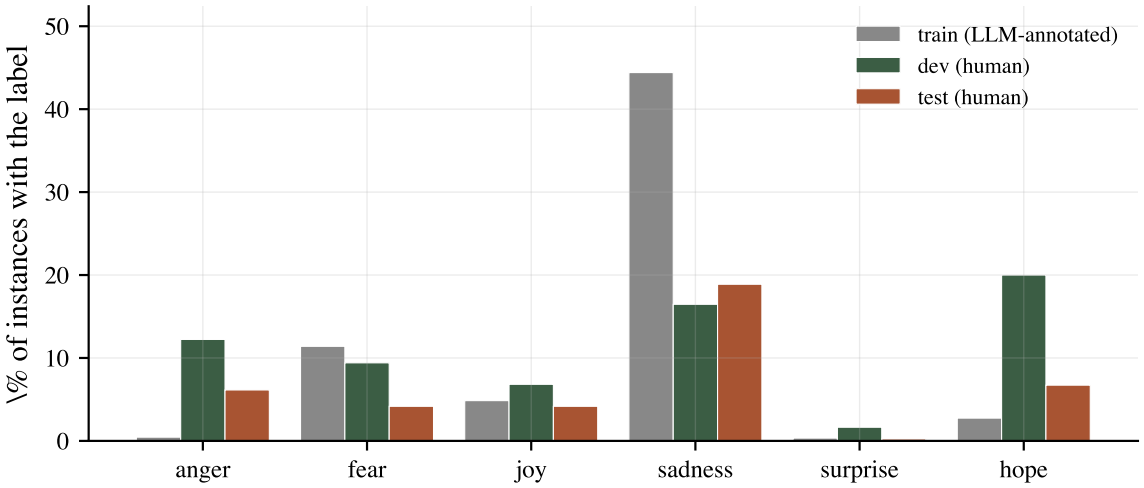

Figure 3: Per-emotion positive rate on the three splits. The LLM-annotated training partition (grey) over-predicts sadness and under-predicts anger and hope, by an order of magnitude in some classes, relative to the human-annotated dev (blue) and test (orange) partitions. Surprise is extremely rare on every split.

under-predicts anger and hope, with the consequence that any model fitted to the training distribution inherits a shifted prior that does not match the human convention used to score the submission.

The finding has a direct architectural consequence. A supervised classifier that minimises the empirical risk on the training partition will, in the absence of explicit reweighting, converge to a posterior that matches the training prior. On a multi-label task with macro-F1, that posterior produces a score catastrophically below the LLM zero-shot baseline that ignores the training data altogether. The agent verified this empirically in the first iteration of the loop. A fine-tuned XLM-RoBERTa-base [19] on the training partition reached only 0.21 macro-F1 on dev, against 0.64 for an early GPT-5.4 [20] few-shot baseline; the gap was predominantly attributable to anger and surprise being predicted zero times by the fine-tuned encoder. We document the specific strategies in Section 6, but the prior takeaway is that the central data-level move of the campaign was to set the training partition aside for any system trained directly on labels and to use it only as a source of unlabelled in-domain text.

3.5. Multi-label structure

A final structural observation is that the multi-label structure of the task is non-trivial but mild. On the dev split the average number of emotion labels per fragment is approximately 0.67 (many fragments are neutral) and approximately fifteen per cent of fragments carry more than one label. On the test split the average is lower at 0.40 because two thirds of test fragments are neutral. The most common label co-occurrences on dev are anger with sadness and hope with sadness, both of which are linguistically natural and do not, in our reading, require any specialised joint-label modelling beyond independent binary classifiers per emotion.

4. Agentic Methodology

This section describes the researcher-in-the-loop methodology in operational detail. We open with a description of the loop and the roles within it, continue with a positioning of the loop on the spectrum of agentic autonomy, formalise the decision protocol with two safety gates, describe the reproducibility artefact, and close with a catalogue of the failure modes we observed during the campaign. The catalogue is, in our view, the central contribution of the section, because it is the empirical substance behind the claim that the researcher-in-the-loop position on the autonomy spectrum is a defensible default for shared-task work with present-day language models.

4.1. The researcher-and-agent loop

The submission was produced inside a tightly coupled loop with two actors. The agent is a Claude Opus 4.7 model [21] exposed to a working directory through a tool-use interface. It can read and edit files, execute shell commands, run Python scripts, query OpenAI and OpenRouter HTTP endpoints, and download artefacts from CodaBench. The researcher is the first author of this paper. The researcher reviews the agent's natural-language proposals, makes the final call on which experiment runs, decides which prediction file is uploaded to CodaBench, and intervenes when a strategy needs to be redirected or abandoned. Figure 4 renders the loop graphically. The loop is a research-specific instance of the human-in-the-loop machine-learning paradigm [22], in which a human supplies the judgments a learning system cannot supply for itself; we adopt it here, rather than the fully autonomous *autoresearch* of Karpathy [6], for the reason argued in Section 4.3, namely that a shared task gives the agent no internal signal that warns it when it is optimising the wrong objective.

A typical iteration of the loop proceeded as follows. The agent inspected the current state of the working directory, read the most recent experimental notes, and proposed a next experiment in natural language, including a hypothesis to test, the resources required, and the metric on which the result would be reported. The researcher accepted, modified or rejected the proposal. In practice, the agent's first proposal was accepted unchanged in the majority of iterations; the researcher's modifications were typically about scope or risk rather than about scientific direction. Once accepted, the agent wrote the code, launched it locally or on a remote GPU, watched the logs, and produced a metric report at the end of the run. Every step of the iteration, including the natural-language reasoning, the tool calls and the shell output, was recorded in a transcript file shared with the working repository. The researcher then reviewed the report and, if the iteration produced a candidate submission, decided whether to upload it. The leaderboard score returned by CodaBench was pasted back into the agent's context, and the next iteration began.

The iterations had highly variable wall-clock duration. The shortest were a handful of minutes, when the agent ran an inference pass on the development set with a small change of prompt. The longest were several hours, when the agent fine-tuned XLM-RoBERTa-large on the training partition and ran the resulting model over the test set. The researcher's work, by contrast, was concentrated at the start and end of each iteration, with the experimental work itself running unattended.

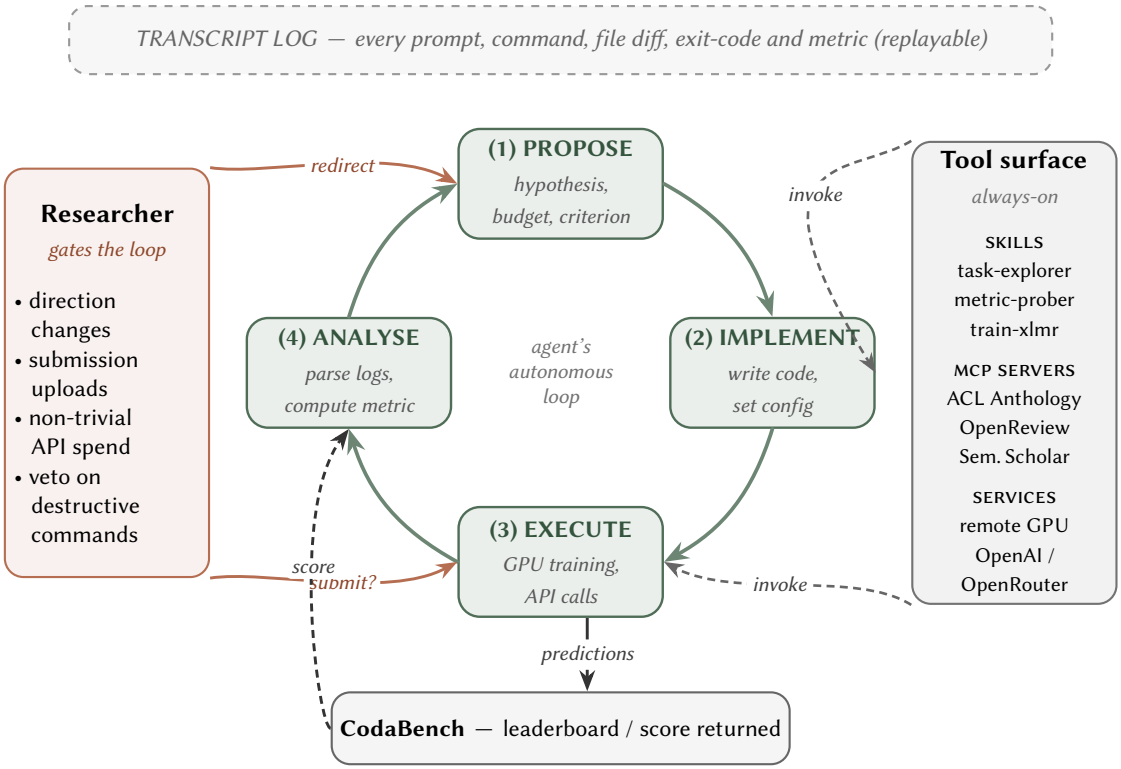

Figure 4: The researcher-and-agent research loop. The agent autonomously proposes, codes, runs and analyses experiments; the researcher gates submissions and direction changes. Every command issued by the agent is recorded in a transcript log that lives alongside the working repository.

Table 1
Division of responsibility between the agent and the researchers across the five stages of a campaign iteration. *Lead* marks the actor who owns the stage; *support* marks the actor who assists.

Stage	Agent	Researchers
Hypothesis formulation	support (suggests alternatives)	**lead** (proposes, frames, prioritises)
Implementation	**lead** (writes the code)	support (debugs)
Execution	**lead** (launches experiments)	support (monitors)
Analysis	support (extracts basic statistics)	**lead** (interprets)
Submission	support (validates the format)	**lead** (approves and uploads)

4.2. Division of labour and the researchers' contribution

Because the boundary between the agent's work and ours is the substance of the methodology, we set it out explicitly. Table 1 assigns, for each of the five stages of a campaign iteration, a *lead* and a *support* role to the agent and to the researchers. The two human researchers are the authors of this paper. The asymmetry of the table is the point: the agent leads the mechanical stages (implementation, execution), the researchers lead the evaluative stages (hypothesis formulation, analysis, submission), and neither actor leads everywhere.

The two stages the researchers lead are not chosen arbitrarily. They are the conjecture and the refutation of the scientific method: proposing a hypothesis worth testing, and judging whether a result counts as one that can be accepted. The agent can accelerate the mechanical span between those two acts, writing the experiment, running it, collecting the numbers, but it does not own either act. The traceable evidence that the researchers actually exercised those two roles is the failure-mode catalogue of Section 4.6: in each documented case the agent's proposal was redirected by a researcher's judgment about what would generalise, and the transcript records the exchange. The submitted system, the parameterless intersection of two models, is itself the product of one such redirection rather than of the

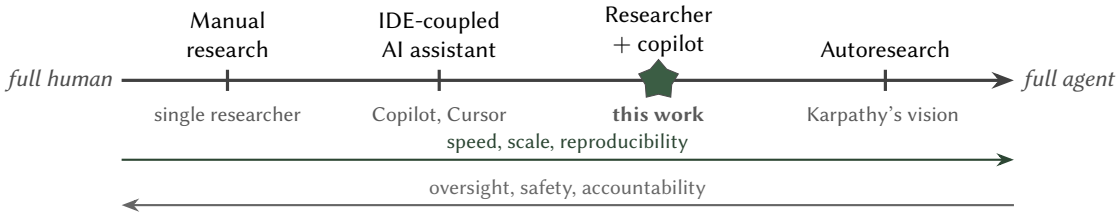

Figure 5: The spectrum of agentic autonomy in research. The horizontal axis ranges from fully manual research at the left to fully autonomous autoresearch at the right. Our position, marked with the star, gives the agent terminal autonomy and reserves submission and direction-change authority for the researcher.

agent's own first instinct, which was a dev-tuned ensemble.

4.3. Position on the autonomy spectrum

The researcher-and-agent loop is not the only way to organise LLM-driven research. Figure 5 sketches a spectrum of autonomy options, from fully manual research at the left to fully autonomous *autoresearch*, in Karpathy's sense, at the right. The intermediate positions are populated. IDE-coupled AI assistants such as Copilot and Cursor live near the left end: the human researcher writes most of the code and uses the AI to accelerate familiar steps. Computer-use agents [5] push further to the right, with the human acting as auditor of multi-step plans. Our position sits in between, closer to autoresearch than to IDE assistance: the agent runs experiments end-to-end without human supervision of individual steps, but the researcher retains authority over externally visible actions (submissions) and direction changes.

We chose this position deliberately. Pure autoresearch on a shared task has, by construction, no internal signal that warns the agent when it is optimising the wrong objective. If the agent misreads what the development set is telling it (and we will report below that ours did, while comparing strategies), no internal feedback will steer it back. A researcher operating with strictly different priors and reading the leaderboard scores at the natural-language level is the cheapest defence against that kind of error. The same argument applies, to a lesser degree, against fully autonomous strategy choices: agents inherit failure modes from their underlying language model, and those failure modes include sunk-cost attachment to a strategy once several hours of compute have been invested. A researcher in the loop is, on those occasions, much cheaper than a wasted day of compute.

4.4. Tools available to the agent and the two safety gates

The agent operated with a generous tool surface and two non-negotiable veto points held by the researcher. Both veto points are operational rather than scientific in character.

The tool surface available to the agent included a sandboxed shell in the working directory, file editing across the entire repository, Python execution, OpenAI and OpenRouter HTTP endpoints, and the public CodaBench download interface. The agent could fine-tune transformer encoders on a remote GPU, run inference in parallel with concurrent HTTP requests, build submission archives in the format expected by CodaBench, and inspect every artefact produced by an experiment. What the agent could not do directly was upload a submission to the leaderboard, push code to GitHub, or commit any change to the released paper. Each of those required a researcher action.

The two safety gates were small in number but operationally decisive. The first gate was the CodaBench upload itself. HISEMOTIONS limited submissions to a small daily quota, and a wasted submission cost an entire day of leaderboard feedback; the researcher reviewed each candidate file before sending it, and on several occasions noticed problems (a malformed CSV header, an unintended row reordering) that the agent had missed. The second gate was external API spend above a small per-iteration threshold; the researcher approved the spend ahead of time when an iteration required it. Within those two gates the agent operated autonomously: hyper-parameter choices, encoder choices, prompt designs, training routines and evaluation infrastructure were all the agent's responsibility.

The looseness of the safety gates is intentional. They were chosen to be rare in operation and decisive when they fired. The researcher made roughly two veto decisions per day during the active part of the campaign, all of them at the upload stage; the agent ran without intervention the rest of the time.

4.5. Reproducibility and the agentic transcript

Every command issued by the agent, every prediction file produced, every model checkpoint saved, and every leaderboard score returned by CodaBench is preserved in the released repository. The agent and researcher dialogue was logged in natural language as a transcript file, which serves the same role for an agentic submission as a hyper-parameter table and a random seed serve for a traditional one. A reader inspecting the transcript can identify, for each non-trivial decision in the campaign, whether the decision was taken by the agent autonomously, proposed by the agent and accepted by the researcher unchanged, or proposed by the agent and modified or rejected by the researcher. We believe this kind of trace is the right reproducibility artefact for agentic submissions in shared tasks, and we recommend that future agentic working notes adopt it as a default.[1] For HISEMOTIONS specifically the transcript runs to several thousand lines and includes the natural-language reasoning that produced the alternative strategies described in Section 6.

Two qualifications on reproducibility follow from the agentic setting and from the models used. First, the coding agent (Claude Opus 4.7) and the two classifiers (GPT-5.4 and GPT-5.5) are proprietary, versioned models behind closed APIs, so an exact bit-for-bit reproduction requires access to the same model versions and is not guaranteed once a provider retires a version. What is reproducible without that access is the finding itself: the prompts, the label-wise intersection rule and the evaluation are released in full, and the central result, that the parameterless intersection of two strong, diverse classifiers outperforms a per-class rule tuned on a small development set, is model-agnostic and can be re-tested with any two such classifiers. Second, the human side of the loop is not anonymous: the two researchers who held the hypothesis-generation and acceptance-criterion roles are the authors, Rodrigo-Ginés and Chamorro-Padial, and the transcript attributes every redirection to a named human action rather than to an unspecified "supervision".

4.6. Observed failure modes

The strongest argument for the researcher-in-the-loop position on the autonomy spectrum is empirical. We observed four qualitatively distinct failure modes during the HISEMOTIONS campaign, two of which would not have been caught by a fully autonomous loop. Each is, in our view, a candidate *a priori* reason for the design we adopted, and we list them at length because the catalogue is itself a contribution to the design of agentic research systems.

The first failure mode is the assumption that more training data is always preferable. The agent's default reaction to a small human-annotated development partition was to combine it with the much larger LLM-annotated training partition under various upweighting schemes. The first such combined run scored a remarkable 0.96 macro-F1 on dev and 0.33 macro-F1 on test. The 63-point gap is the unmistakable signature of evaluation-set leakage: the dev partition had been used both for training and for evaluation, and the score on dev was an estimate of memorising capacity rather than of generalisation. A fully autonomous loop without an external researcher would have noticed the gap (the test score is also returned by CodaBench) but the agent's transcript at that point already contained a proposal to "increase the dev upweighting from 5 to 10 to push dev higher", which would have made the leakage worse rather than better. The researcher caught the failure mode by asking, in natural language, whether the agent was confident that combining training and evaluation data with upweighting was a defensible decision. The agent then noticed the leakage on its own and reverted to a clean separation.

The second failure mode is dev overfitting through per-class tuning. On a task with a small (425-instance) development set and six classes evaluated independently, any procedure that parameterises the predictor through dev macro-F1 has effectively six free parameters tuned on a small sample. The agent's

[1] Code, predictions, the agent transcript and the reproduction recipe are available at https://github.com/franfj/HISEMOTIONS.

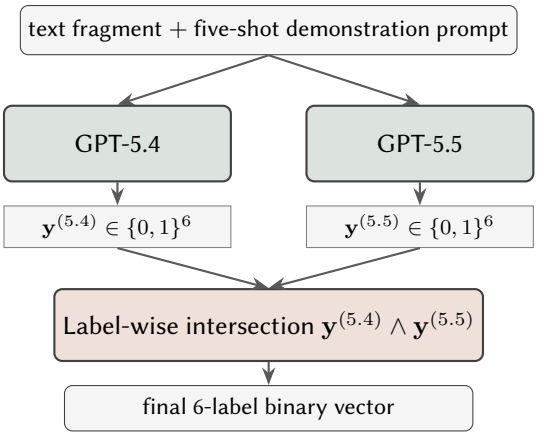

Figure 6: System architecture. Each text fragment, together with the same five-shot demonstration prompt drawn from the development set, is sent to GPT-5.4 and to GPT-5.5 independently. Each model returns a binary vector over the six emotion labels. The submitted prediction is the label-wise logical AND of the two vectors.

proposal on the ensemble exploration phase was to tune a per-class merge rule across three diverse models, picking, for each emotion independently, whichever of `any`, `majority`, `all`, or `single-model` produced the best class-conditional F1 on dev. The procedure reached 0.728 macro-F1 on dev, the highest in the campaign. Test macro-F1 was 0.539, a regression of approximately 0.20 that was almost entirely attributable to per-class tuning. The corresponding parameterless intersection rule scored 0.649 on dev, fifteen points lower, but 0.5775 on test, four points higher. We discuss the dev-set-size effect formally in Section 7; the relevant methodological point is that the agent's prior was that more flexible strategies are preferable, and the empirical evidence rejected that prior on this specific task family.

The third failure mode is conservative model substitution. On several occasions the agent attempted to use a newer language model (GPT-5.5) on the assumption that the newer model's larger capability would translate into a better score. On HISEMOTIONS the substitution did not regress sharply, but it also did not improve the score by as much as the agent had expected: GPT-5.5 alone scored 0.5564 macro-F1 on test against 0.5338 for GPT-5.4. The agent's instinct was to replace GPT-5.4 entirely with the newer model, but the researcher noticed that the per-class profile of the two models was different in interesting ways, with GPT-5.5 stronger on hope and joy and GPT-5.4 stronger on sadness, and proposed to keep both models in play rather than to substitute. The pure intersection of the two binary outputs emerged from that proposal and turned out to be the best strategy of the campaign.

The fourth failure mode is implicit assumption transfer. The agent's prompt designs often inherited assumptions from the first version it wrote, which were not always appropriate for later experiments. The clearest instance during this campaign was a chain-of-thought variant of the GPT-5.4 prompt: the agent believed, on prior, that explicit reasoning would improve the score, and observed an improvement on dev that did not transfer to test. The mode is the closest analogue, in the agentic setting, of the silent bug in a long-running training loop: it cannot be caught by reading any single command in isolation, only by inspecting the strategy at the level at which the agent thinks about it.

5. System Description

The submitted system is a label-wise intersection of two strong prompted language models (Figure 6). We describe each component, then the intersection rule, then the practical reasons we did not include a third model in the intersection.

5.1. GPT-5.4 zero-shot prompt with curated demonstrations

The first component is a zero-shot prompt to GPT-5.4 [20] with sixteen curated in-context demonstrations. The system prompt is in Spanish and describes the task as identifying which of the six emotions are expressed in a fragment from a sixteenth-century Spanish letter, with the option of returning "none" if no emotion is detected. The user message contains the fragment in plain text and the model is asked to respond with a comma-separated list of emotion labels. The in-context demonstrations are sixteen instances drawn from the development partition, chosen by the agent to cover all six emotions and a handful of edge cases (courtesy formulas, formal salutations, neutral metadata sentences such as letter headers). The prompt explicitly instructs the model that historical Spanish contains archaic spellings, that "VM" abbreviates "Vuestra Merced" and is not by itself emotional, and that courtesy formulas like "beso las manos" are conventional rather than joyful. The temperature is set to zero to make the predictions deterministic, and the model is prompted in JSON mode where it is available so that the output is mechanically parseable.

We map the comma-separated label list to a binary vector $\hat{\mathbf{y}}^{(54)} \in \{0,1\}^6$ with one entry per emotion. The same prompt is used for both the development set and the test set. The component reaches 0.642 macro-F1 on dev and 0.534 macro-F1 on test. Per-class profiles are reported in Section 6; the headline observation is that GPT-5.4 is strongest on sadness and anger and weakest on hope, where it has high recall but low precision because it occasionally reads forward-looking polite phrases as expressions of hope.

5.2. GPT-5.5 zero-shot prompt with the same demonstrations

The second component is the same prompt template applied to GPT-5.5. We make no attempt to re-tune the prompt for the newer model: the in-context demonstrations, the system prompt and the output format are identical. The component reaches 0.650 macro-F1 on dev and 0.556 macro-F1 on test, slightly stronger than GPT-5.4 on both splits but with a different per-class profile. GPT-5.5 is stronger on joy (0.789 test F1 against 0.720 for GPT-5.4) and stronger on hope (0.481 test F1 against 0.368 for GPT-5.4); it is slightly weaker on anger and roughly equal on the rest. We map the GPT-5.5 output to a binary vector $\hat{\mathbf{y}}^{(55)} \in \{0,1\}^6$ in exactly the same way as the GPT-5.4 output. The two models behave sufficiently differently per class that their disagreement carries genuine signal, which is the property the intersection rule exploits.

5.3. Label-wise intersection

The submitted prediction is the elementwise logical AND of the two binary vectors:

$$\hat{y}_{i,e} = \hat{y}^{(54)}_{i,e} \wedge \hat{y}^{(55)}_{i,e} \quad \text{for each instance } i \text{ and each emotion } e. \tag{1}$$

A label is positive in the submission if and only if both models predict it. The rule has zero free parameters: there is no threshold to tune, no per-class weighting, no merge rule selected on dev. We deliberately considered and rejected several parametrised generalisations while comparing strategies: a per-class merge rule tuned on dev (which dev-overfitted, as described in Section 6), a soft-vote average-then-threshold scheme that requires at least one new hyper-parameter, and a stacked logistic regression on top of the two model outputs that was discarded for the same reason. In every case the parameterless intersection beat the parametrised alternative on the held-out test set.

5.4. Why two models and not three

The natural extension is to intersect three models rather than two. We considered Claude Opus 4.7 [21] and LLaMA-3.3-70B as candidate third models. Both produced empty predictions on the development set in the agent's first pass with a uniform prompt, with a vector of all zeros for every fragment; the issue was prompt-format related and the agent did not have the development time to debug it before the deadline. The absence of a third strong, diverse model in the intersection is the principal limitation of

Table 2
Strategies the agent explored that produced both a development and a test score, in chronological order. Dev and Test are macro-F1 with respect to the six-emotion label vector. "Notes" summarises the lesson the agent took from the strategy. The bold row is the official entry.

#	Strategy	Dev	Test	Notes
1	XLM-R-base "combined", dev $\times 5$	0.96	0.33	evaluation-set leakage
2	GPT-5.4 zero-shot 5-shot	0.620	0.522	first competitive submission
3	GPT-5.4 chain-of-thought	0.550	0.465	negative result; reasoning hurt
4	Synthetic 16th c. Spanish + BETO	1.00	0.318	augmented memorisation; rejected
5	GPT-5.4 alone (final prompt, 16-shot)	0.642	0.534	strong single model
6	GPT-5.5 alone (same prompt)	0.650	0.556	stronger single model
7	Per-class ensemble of three LLMs	0.728	0.539	dev overfit; $\Delta = -0.189$
8	GPT-5.4 with meta-prompted hard cases	0.701	0.498	dev overfit; $\Delta = -0.203$
9	**GPT-5.4 $\cap$ GPT-5.5 (intersect.)**	**0.649**	**0.578**	**official, rank 1**
10	GPT-5.4 $\cup$ GPT-5.5 (union)	0.645	0.522	union over-predicts; rejected

the submitted system, and we expect that closing the gap with the leading systems on the leaderboard would primarily require extending the intersection to a third LLM with a different inductive bias from GPT-5.4 and GPT-5.5. Section 7 returns to this point.

6. Strategies, Results, and the 2-LLM Intersection Finding

We list, in chronological order of the agent's exploration, the strategies that were implemented and either submitted to CodaBench or evaluated locally. We then report the central empirical observation that motivated this paper: on a task with a small human-annotated development set, the parameterless label-wise intersection of two strong prompted LLMs outperforms a per-class-tuned ensemble of three on the held-out test set, despite scoring lower on the development set. The pattern is visible directly in the dev-test gaps reported in Table 2 and in Figure 7.

6.1. Strategies tabulated

Table 2 summarises the agent's exploration. We include strategies that were submitted to CodaBench, strategies that were prepared but never submitted because the daily quota was exhausted, and strategies that were evaluated only locally. The bold row is the official entry.

The table tells a campaign-level story that is easy to summarise. The first phase of the campaign (rows 1 to 4) was an exploration of the standard supervised-learning toolbox: a fine-tuned encoder, a chain-of-thought prompt, and synthetic data augmentation. Every strategy in that phase either underperformed the simple GPT-5.4 zero-shot baseline (rows 3 and 4) or suffered from evaluation-set leakage (row 1). The strongest result of the first phase was the GPT-5.4 zero-shot baseline at 0.522 test macro-F1.

The second phase (rows 5 to 10) explored ensembles, and the agent prepared three substantive changes. The first was a per-class dev-tuned ensemble across GPT-5.4, GPT-5.5 and a meta-prompted GPT-5.4 variant; it reached 0.728 on dev, the highest of the campaign, but collapsed to 0.539 on test. The second was a targeted re-labelling: GPT-5.5 was asked to re-label fragments where GPT-5.4 had returned no emotion, on the hypothesis that it would catch surprise and hope cases GPT-5.4 had missed; on the 267 fragments examined it returned "neither" on every one, a null result we record honestly as a negative finding. The third was a meta-prompted GPT-5.4 carrying explicit false-negative and false-positive demonstrations for surprise and hope; it reached 0.701 on dev but 0.498 on test, again a substantial overfit. With three regressing candidates, the agent and the researcher turned to the simplest strategy not yet submitted: the label-wise intersection of GPT-5.4 and GPT-5.5. The intersection scored 0.649 on dev, lower than the per-class ensemble, and 0.578 on test, substantially higher than any other

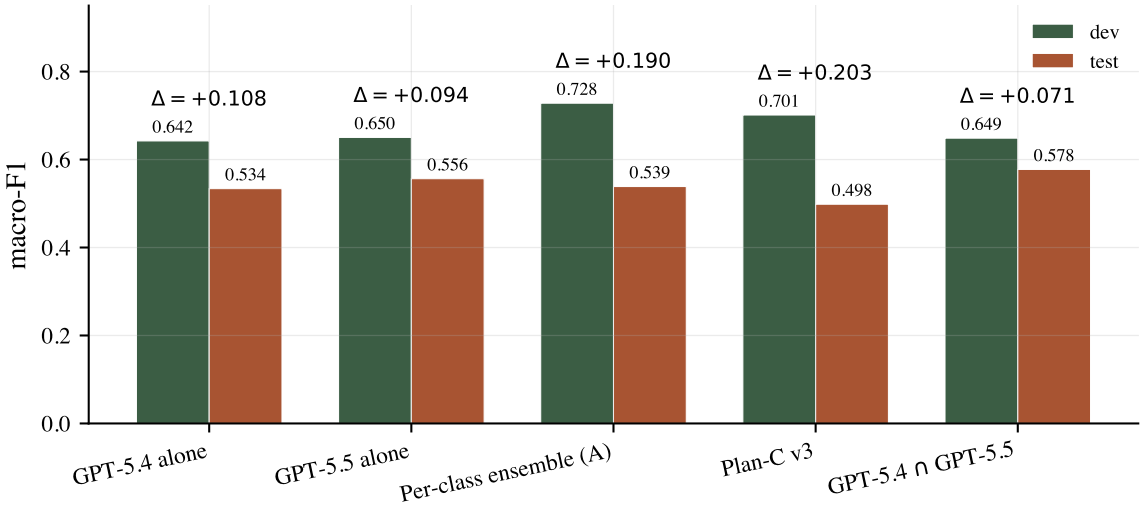

Figure 7: Dev and test macro-F1 for the five strongest alternative strategies. The two parametrised strategies (per-class ensemble of three LLMs and the meta-prompted variant) have dev-test gaps near 0.20, indicating substantial overfit to the small development set. The parameterless intersection of GPT-5.4 and GPT-5.5 has a smaller gap and the highest test score.

candidate.

6.2. The dev-test gap signature

The empirical pattern that connects the alternative strategies is best read off the dev-test gap rather than the dev or test score in isolation. Figure 7 reports both numbers for the strongest strategies. The pattern is qualitatively unambiguous. The two parameterless strategies (GPT-5.4 alone, GPT-5.5 alone, and the intersection) have small dev-test gaps (between 0.07 and 0.10). The two parametrised strategies (the per-class ensemble of three LLMs and the meta-prompted variant) have very large dev-test gaps (0.19 and 0.20). The intersection has the smallest gap of any submitted strategy and the highest test score, despite scoring lower on dev than either of the parametrised strategies.

The pattern is, in our reading, a development-set-size effect. On a task with macro-F1 averaged across six labels and a 425-instance dev set, a per-class merge rule has effectively six free parameters (one rule per class) tuned on a small sample. The variance of the dev macro-F1 across alternative merge rules is not negligible, and the procedure that maximises dev macro-F1 is likely to pick a combination of class-conditional rules that is optimal on the dev sample but not optimal on the population. The parameterless intersection has zero free parameters and is therefore immune to this form of overfitting. The cost of the intersection is that it is more conservative on classes where the two models disagree, but on the test set the conservativeness is rewarded rather than punished.

6.3. Per-emotion intersection profile

Figure 8 reports per-emotion test F1 for the two single-model baselines (GPT-5.4 and GPT-5.5), the meta-prompted variant variant, and the intersection. The intersection beats either single model on five of the six emotions and ties on joy. The per-class gains are most pronounced on surprise (where both models occasionally over-predict, and the intersection removes the false positives) and on hope (where GPT-5.4 is imprecise and GPT-5.5 is conservative, with the intersection capturing the high-precision agreement subset). On sadness the intersection is slightly weaker than GPT-5.5 alone, because GPT-5.4 and GPT-5.5 disagree on a non-trivial subset of sad fragments and the intersection removes the predictions of the disagreeing instances. The aggregate effect across the six emotions is positive in macro-F1 because the gains on hope and surprise outweigh the loss on sadness.

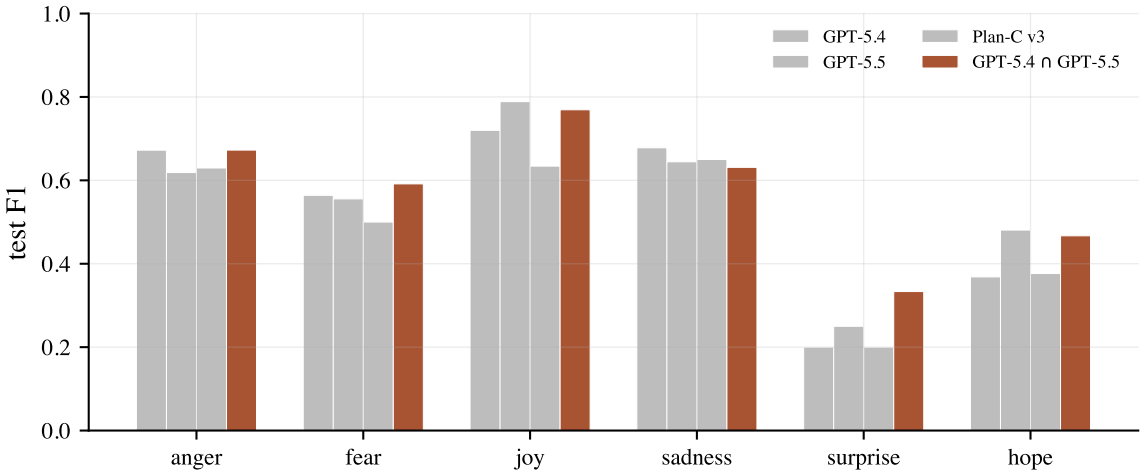

Figure 8: Per-emotion test F1 for the alternative candidates. The intersection of GPT-5.4 and GPT-5.5 (orange) wins or ties on every emotion except sadness, with the largest gains on hope and surprise. The aggregate macro-F1 of the intersection (0.578) is higher than that of either single model.

Table 3
Official leaderboard slice for HISEMOTIONS at IberLEF 2026. The bold row is our submission. Macro-F1 is averaged over the six emotion labels by the official scoring program.

Rank	Team	Macro-F1
1	**ours**	**0.5775**
2	carlosdf	0.5300
3	Sinai	0.4500
6	explainators	0.3100

6.4. What the intersection finding implies

The empirical pattern has a generalisable consequence. On multi-label tasks with macro-averaged metrics and small human-annotated development sets, per-class tuning is a high-variance procedure whose dev score is a poor estimator of its test score. The label-wise intersection of two strong classifiers is a parameterless alternative whose dev score is a faithful estimator of its test score, because there are no free parameters to be tuned on the dev sample in the first place. We do not claim the intersection is universally optimal: in cases where the two classifiers agree on every label the intersection collapses to the single classifier, and in cases where one classifier is much weaker than the other the intersection inherits the weaker classifier's recall floor. We do claim that the intersection is the right default to consider before investing in a parametrised merge rule, particularly when the development set is small enough that variance dominates bias on parametrised procedures.

6.5. Final score and official ranking

Our official submission, row 9 in Table 2, scored 0.5775 on the macro-F1 metric of the public test set, and achieved the highest test score on the leaderboard at submission time. Table 3 reports the relevant slice of the official leaderboard: the top three submissions, our submission, and the last submission.

6.6. Error analysis

The intersection's per-emotion F1 on the public test partition is reported in Table 4. Macro-F1 is 0.5775 as a per-class average, but the contribution per class is unequal: *joy* and *anger* are the strongest, both above 0.65, while *surprise* is the weakest at 0.33. Two structural factors explain the surprise result. First, only seven test fragments carry a positive *surprise* label, so a single misclassification moves the class

Table 4
Per-emotion F1 of the submitted intersection on the public test partition.

Emotion	Test F1	Test support
anger	0.673	53
fear	0.592	36
joy	0.769	36
sadness	0.631	163
surprise	0.333	2
hope	0.467	58
macro avg.	**0.578**	—

F1 by approximately 0.07. Second, the historical register of the texts uses surprise markers that are infrequent in the demonstration prompt: stylised exclamations, *admiratio* formulae, and stock phrases that do not translate cleanly to the modern LLM's prior over "surprise" as a discrete emotion.

Hope is the second-weakest class at F1 0.467. Both component classifiers (GPT-5.4 and GPT-5.5) over-predict *hope*: the gold partition has 58 *hope*-positive fragments, while GPT-5.4 alone predicts 212 and GPT-5.5 alone predicts 152. The intersection brings the prediction count down to 139, which is closer to gold but still high. The empirical signature is consistent with a prior in both LLMs that any expression of expectation, divine will or reference to a future better state should count as *hope*; a fraction of the intersection's hope-positives on test are stylised closing formulae of historical letters that are conventional and not actually hopeful.

Read qualitatively, the intersection's outputs fall into three recognisable kinds. The true positives concentrate where the emotion is explicit in the modern sense, declarations of grief, of fear for a relative's safety, or of joy at a reunion, which both classifiers label and the intersection therefore keeps. The false positives are dominated by register: conventional epistolary formulae such as devout closings and formulaic well-wishing, which a present-day model reads as hope or joy but which carry no emotional charge in the period convention. The false negatives are the mirror image, period-specific markers of surprise and indignation that fall outside a modern model's prior and are caught by at most one of the two classifiers, so the intersection drops them. The residual error is therefore better described as a cross-temporal register gap than as a failure of the intersection rule itself.

7. Discussion: Lessons for Agentic Research

Three transferable lessons came out of this campaign. We list them in the order in which we believe they generalise, from the most robust to the most task-specific.

The first lesson is that an agentic research loop should characterise the development set's statistical properties before optimising on it. The HISEMOTIONS development set is small enough (425 instances, six classes) that any procedure with more than a handful of free parameters tuned on it is at risk of overfitting in a way that is invisible from the dev score itself. The agent that ran our campaign read the dataset's README and inspected the per-class frequencies on the first iteration, but it did not, on the ensemble exploration phase, ask the question "how many free parameters am I tuning on dev, and what is the variance of the dev score under those parameters". A simple experiment, namely running the per-class merge-rule procedure under several alternative random seeds and reporting the variance of the dev score across seeds, would have revealed the dev-set-size effect within a single iteration of the loop. We now consider this kind of dev-set characterisation the default first action for any agentic submission to a multi-label task with a small dev set, on a par with reading the dataset's README. The lesson is methodological and not specific to HISEMOTIONS: any task with a small dev partition is at risk of the same overfitting failure, and the cost of characterising the dev variance is far smaller than the cost of submitting a regressing strategy under the daily-quota constraint.

The second lesson is that the parameterless label-wise intersection of two strong, diverse classifiers is

the right default ensembling rule for multi-label tasks with limited development data. The intersection has zero free parameters and is therefore immune to dev overfitting; it relies on the diversity between the two classifiers for its gains, which means that the two classifiers must be drawn from genuinely different families to be useful. In our setting GPT-5.4 and GPT-5.5 provided enough per-class diversity to make the intersection beat either single model on five of six emotions, but the diversity is far from guaranteed for arbitrary pairs of LLMs and should be verified empirically on the dev set before committing to the strategy. The lesson generalises naturally: any multi-label classification setting with a sufficiently small development set should consider the intersection of two strong classifiers as a baseline before investing in a tuned merge rule, and any tuned merge rule should be benchmarked against the parameterless intersection on the test set.

The third lesson is the case in favour of researcher-in-the-loop agentic research. Three of the four documented failure modes in Section 4.6, namely the assumption that more training data is always preferable, dev overfitting through per-class tuning, and conservative model substitution, would not have been caught by a fully autonomous loop without an external researcher. The intervention in each case took the form of a single natural-language question from the researcher: *can you defend why this strategy should generalise.* In every case the agent then noticed the failure on its own and revised. A fully autonomous loop with no external researcher would, on the evidence of this campaign, have spent additional days on dev overfitting and on combined-mode leakage; the cost of the researcher's interventions was, in aggregate, perhaps an hour of attention per day. We view the researcher-and-agent model as strictly preferable to fully autonomous autoresearch for shared-task work with present-day language models, and we expect this preference to weaken only as language models become better at metacognition over their own training-time priors.

8. Limitations

Three limitations bound the claims of this paper. The first is in the data. The training partition is LLM-annotated while the development and test partitions are human-annotated, and the two differ enough that a model fine-tuned on the training partition never became competitive (an XLM-RoBERTa-base baseline reached only 0.21 macro-F1 on dev, against 0.64 for an early prompted baseline). Our system therefore sidesteps the training partition rather than using it, and a method that could exploit the LLM-annotated data without inheriting its bias remains open; our results should be read as specific to the prompted-LLM regime that the annotation mismatch forced.

The second limitation is the one our own central finding turns on. With a 425-instance development set and six independent classes, the development score is a high-variance estimate, and any procedure tuned on it, including the ones we ourselves tried, is exposed to overfitting that is invisible from the development number alone. We report the intersection as the robust default, but the same small-sample fragility that defeated the tuned ensembles also limits the precision of every dev-based comparison in the paper; the test-set differences of a few points between nearby strategies should be read with that variance in mind.

The third limitation is reproducibility under closed models, as discussed in Section 4. The agent and the two classifiers are proprietary and versioned; the released prompts, intersection rule and transcript make the *finding* reproducible with any two strong classifiers, but they do not make the exact scores reproducible once the specific model versions are withdrawn.

9. Conclusion

We described an agentic submission to HISEMOTIONS at IberLEF 2026 in which a coding agent ran the experimental loop under researcher supervision. The submitted system is the parameterless label-wise intersection of two strong prompted language models, GPT-5.4 and GPT-5.5, with an identical sixteen demonstration prompt template. The system scored 0.5775 macro-F1 on the public test set. Three parametrised alternatives that the agent compared, a per-class dev-tuned ensemble of three LLMs, a

targeted re-labelling pass, and a meta-prompted variant with explicit hard-case demonstrations, all overfitted the small development partition and underperformed the parameterless intersection on the held-out test set. The pattern is a development-set-size effect that should generalise to any multi-label task with limited human-annotated dev data, and the intersection is the default we recommend before investing in a parametrised merge rule. The methodological contribution of the paper is the researcher-and-agent loop in which the campaign was run, and the catalogue of failure modes we observed during it; we release the data, the code and the agent transcript so that other participants in agentic shared-task work can adopt or contest the working model.

Declaration on Generative AI

This submission used generative AI as part of the system itself and as part of the experimental loop, in two operationally distinct roles. The first role is methodological. Claude Opus 4.7 (Anthropic) acted as a coding agent that autonomously implemented experiments, ran them on remote GPUs, parsed logs and updated its strategy in light of leaderboard feedback. The agent was given the task description, the raw data, GPU access and API credentials, and was responsible for the full life cycle of every experiment described in Section 6. Every tool call and every natural-language reasoning step taken by the agent was logged in the transcript file released with the code. The researcher (the first author) reviewed the agent's proposals, made the final call on which experiment ran, decided which prediction file was uploaded to CodaBench, and intervened on strategic-direction decisions. The second role is system-level. GPT-5.4 (OpenAI) and GPT-5.5 (OpenAI) were used at inference time as the two components of the submitted intersection, via the `chat.completions` API in JSON mode. The agent considered Claude Opus 4.7 and LLaMA-3.3-70B as candidate third components of the intersection but did not include them in the final submission for the reasons given in Section 5.

References

[1] A. Sarymsakova, P. Martín-Rodilla, E. Martínez-Cámara, L. A. Ureña-López, Overview of HISE-MOTIONS at IberLEF 2026: Historical Text-Based Emotion Detection in Early Modern Spanish Correspondence, Procesamiento del Lenguaje Natural 77 (2026).

[2] A. Bonet-Jover, J. Á. González-Barba, L. Chiruzzo, Overview of IberLEF 2026: Natural Language Processing Challenges for Spanish and other Iberian Languages, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2026), co-located with the 42nd Conference of the Spanish Society for Natural Language Processing (SEPLN 2026), CEUR-WS.org, 2026.

[3] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, A. Anandkumar, Voyager: An open-ended embodied agent with large language models, Trans. Mach. Learn. Res. 2024 (2024). URL: https://arxiv.org/abs/2305.16291.

[4] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, J. Tang, Agentbench: Evaluating llms as agents, in: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024, 2024. URL: https://arxiv.org/abs/2308.03688.

[5] Anthropic, Developing a computer use agent, https://www.anthropic.com/research/developing-computer-use, 2024.

[6] A. Karpathy, Autonomous research with llm agents ("autoresearch"), https://karpathy.ai/, 2026. Open-source repository sketching an autonomous research loop driven by LLM agents.

[7] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, J. Gao, Large language models: A survey, arXiv:2402.06196, 2024.

[8] S. M. Mohammad, F. Bravo-Marquez, M. Salameh, S. Kiritchenko, Semeval-2018 task 1: Affect in tweets, in: Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018, Associa-

tion for Computational Linguistics, 2018, pp. 1–17. URL: https://doi.org/10.18653/v1/s18-1001. doi:10.18653/V1/S18-1001.

[9] F. M. P. del Arco, C. Strapparava, L. A. U. López, M. T. M. Valdivia, Emoevent: A multilingual emotion corpus based on different events, in: Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020, European Language Resources Association, 2020, pp. 1492–1498. URL: https://aclanthology.org/2020.lrec-1.186/.

[10] S. H. Muhammad, N. Ousidhoum, I. Abdulmumin, S. M. Yimam, F. M. Plaza-del Arco, A. F. Aji, et al., SemEval-2025 Task 11: Bridging the Gap in Text-Based Emotion Detection, in: Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 2558–2569. URL: https://aclanthology.org/2025.semeval-1.327/.

[11] S. H. Muhammad, F. M. Plaza-del Arco, A. F. Aji, et al., BRIGHTER: Bridging the gap in human-annotated textual emotion recognition datasets for 28 languages, in: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 8895–8916. URL: https://aclanthology.org/2025.acl-long.436/.

[12] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, N. D. Lawrence, Dataset Shift in Machine Learning, The MIT Press, 2009. doi:10.7551/mitpress/9780262170055.001.0001.

[13] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recognition 45 (2012) 521–530.

[14] S. Riedel, L. Yao, A. McCallum, Modeling relations and their mentions without labeled text, in: Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III, Lecture Notes in Computer Science, Springer, 2010, pp. 148–163. URL: https://doi.org/10.1007/978-3-642-15939-8_10. doi:10.1007/978-3-642-15939-8_10.

[15] Z. C. Lipton, C. Elkan, B. Naryanaswamy, Optimal thresholding of classifiers to maximize F1 measure, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), 2014, pp. 225–239.

[16] M.-L. Zhang, Z.-H. Zhou, A review on multi-label learning algorithms, IEEE Transactions on Knowledge and Data Engineering 26 (2014) 1819–1837.

[17] T. Schmidt, K. Dennerlein, C. Wolff, Towards a corpus of historical german plays with emotion annotations, in: 3rd Conference on Language, Data and Knowledge, LDK 2021, Zaragoza, Spain, September 1-3, 2021, OASIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 9:1–9:11. URL: https://doi.org/10.4230/OASIcs.LDK.2021.9. doi:10.4230/OASICS.LDK.2021.9.

[18] R. Sprugnoli, S. Tonelli, A. Marchetti, G. Moretti, Towards sentiment analysis for historical texts, Digital Scholarship in the Humanities 31 (2016) 762–772.

[19] 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.

[20] OpenAI, GPT-5.4 Thinking System Card, https://openai.com/index/gpt-5-4-thinking-system-card/, 2026.

[21] Anthropic, Claude opus 4.7: System card, https://www.anthropic.com/news/claude-opus-4-7, 2026.

[22] S. Amershi, M. Cakmak, W. B. Knox, T. Kulesza, Power to the people: The role of humans in interactive machine learning, AI Magazine 35 (2014) 105–120. doi:10.1609/aimag.v35i4.2513.

[23] Anthropic, Introducing the Model Context Protocol, https://www.anthropic.com/news/model-context-protocol, 2024.

[24] I. Gusev, academia_mcp: Tools for automatic scientific research, https://github.com/IlyaGusev/academia_mcp, 2025. Model Context Protocol server exposing scientific-research tools, including the ACL Anthology bibliography.

[25] F.-J. Rodrigo-Ginés, J. Chamorro-Padial, openreview-mcp: A Model Context Protocol server for querying peer review at scale (v1.1), Zenodo, 2026. URL: https://doi.org/10.5281/zenodo.19772807.

doi:10.5281/zenodo.19772807.
[26] J. Kuo, semanticscholar-MCP-Server: A Model Context Protocol server for the semantic scholar api, https://github.com/jackkuo666/semanticscholar-mcp-server, 2025. Model Context Protocol server exposing the Semantic Scholar academic graph.

A. Tool Surface: Skills and MCP Servers

This appendix documents the tool surface that the coding agent operated with throughout the campaign. We separate the surface into three layers: project-scoped *skills* that encode reusable methodological guidance, MCP servers that expose external research databases as structured tools, and lower-level shell, Python and HTTP capabilities. The complete set of skills and MCP server configurations ships with the released repository.

A.1. Skills

The agent loaded a small library of project-scoped skills written in the Claude Code skill format. Each skill is a markdown file with front matter and a body of natural-language guidance that the agent reads when it detects relevance. The skills below were the ones that drove the actual research workflow on this task, from data exploration to final submission, and were reused across every IberLEF 2026 task we participated in.

- `iberlef-task-explorer`: opens a fresh task by reading its description, downloading the data and producing an exploratory report (label distributions, length distributions, train/dev/test mismatch flags, language balance, expected difficulty). For HISEMOTIONS this skill surfaced the LLM-annotated training set bias on the first iteration, which dominated the rest of the campaign.
- `iberlef-metric-prober`: characterises the official scoring metric empirically before optimising it. Runs ablations against a placeholder submission and reports which features of the output the metric actually rewards.
- `iberlef-baseline`: runs a battery of fast baselines (TF-IDF + logistic regression, XLM-RoBERTa-base, GPT-5.4 zero-shot prompt) on a new task within minutes. On HISEMOTIONS it produced the calibration that justified discarding the LLM-annotated training set entirely.
- `iberlef-train-xlmr`: fine-tunes XLM-RoBERTa-base or large on a task-specific objective, with the checkpoint-saving discipline that protects against remote-GPU instance termination. On HISEMOTIONS the skill fired but the resulting fine-tuned model never beat the prompted baseline.
- `iberlef-llm-prompt`: designs, tests and iterates LLM prompts for a task. Includes JSON-mode output handling and prompt validation against a held-out slice. On HISEMOTIONS the skill produced the static five-shot prompt used by the GPT-5.4 baseline and the chain-of-thought variant.
- `iberlef-ensemble`: combines per-strategy predictions into ensemble submissions, including Borda counts, score-level fusions, and the label-wise intersection rule that produced our final HISEMOTIONS submission.
- `iberlef-error-analyser`: produces per-class and per-instance error reports. On HISEMOTIONS it surfaced the over-prediction of *hope* and the under-prediction of *joy* that the chain-of-thought variant tried, and failed, to fix.
- `iberlef-codabench-submit`: packages a prediction file in the format the task expects, runs format validation, and produces the submission archive for the researcher to upload.

A.2. MCP servers

The agent communicated with three Model Context Protocol (MCP) servers [23] during the experimentation phase. The servers were used to ground the model and technique choices in actual published work rather than the agent's training-cutoff memory, and to short-circuit the tendency of LLM agents to reproduce a recipe they have seen rather than the right recipe for the task at hand.

The **ACL Anthology MCP server** [24] exposes the ACL bibliography as a queryable resource. On HISEMOTIONS the agent used it to retrieve prior work on multi-label emotion classification with rare-class imbalance, which led to the decision to weight the macro-F1 components explicitly when tuning the static-shot prompt rather than relying on the model to infer the metric.

The **OpenReview MCP server** [25] is the first author's own contribution to the MCP ecosystem and exposes peer-review and submission metadata for NeurIPS, ICLR and adjacent venues. On HISEMOTIONS the agent queried it for recent work on calibrating language models to small human-annotated dev sets and for the literature on distribution-shift robustness in classification.

The **Semantic Scholar MCP server** [26] provided citation-graph traversal that the agent used to expand from a small seed set of papers (e.g. the original Plutchik emotion-wheel paper for the dimension layout, and our own prior work on Spanish text classification for the encoder choice) to a broader collection of related work via forward and backward citations.

A.3. Lower-level capabilities

The agent additionally had a sandboxed shell, Python execution, file editing across the repository, OpenAI and OpenRouter HTTP endpoints, and the public CodaBench download interface. The agentic transcript file released with the code records every tool call and every natural-language reasoning step, which serves the same role for an agentic submission as a hyper-parameter table and a random seed serve for a traditional one.

B. Concrete Prompts

This appendix lists the concrete prompts used in the submitted system. We include them verbatim so that the system can be reproduced by anyone with access to OpenAI's GPT-5.4 and GPT-5.5 models.

B.1. GPT-5.4 / GPT-5.5 emotion classification prompt

The same system prompt was used for both GPT-5.4 and GPT-5.5 (only the model identifier in the API call differed). Each call classifies a single text fragment.

```
You are an expert in detecting emotions in 16th-17th century Spanish letters (Early Modern Spanish
    ↪ ). Given a text fragment from a historical Spanish letter, identify which emotions are
    ↪ expressed. The possible emotions are: anger, fear, joy, sadness, surprise, hope. A text
    ↪ can express multiple emotions or none at all.
Important guidelines: These are historical texts with archaic spelling and grammar. "VM" = "
    ↪ Vuestra Merced" (Your Grace), a form of address. Look for emotional content, not just
    ↪ emotional words. Courtesy formulas like "beso las manos" are conventional, not necessarily
    ↪  joyful. Letters discussing death, illness, or loss often express sadness and/or fear.
    ↪ Requests for help or divine intervention often express hope. Complaints about injustice or
    ↪  mistreatment often express anger. Texts can be neutral (no emotion): this is common.
Here are examples of how to classify: {examples_str}.
Respond with ONLY the emotion labels separated by commas, or "none" if no emotions are detected.
    ↪ Do not explain your reasoning.
```

The placeholder `{examples_str}` is filled with five demonstration instances drawn from the human-annotated dev set, formatted as `Text: "..."\nEmotions: ...`. The user message contains the test fragment prefixed by `Classify the emotions in this text:`. Both models are called via the JSON-free `chat.completions` API; outputs are parsed by tokenising the comma-separated label list.

B.2. Label-wise intersection rule

The submission combines the binary outputs of the two prompted models per emotion label. For each test instance and each emotion e,

$$\hat{y}_e = \mathbb{1}\big[\hat{y}_e^{(5.4)} = 1 \text{ and } \hat{y}_e^{(5.5)} = 1\big].$$

No per-class thresholds, no learned weights, no dev-set tuning.

B.3. Agent system prompt

The Claude Opus 4.7 coding agent was operated with the standard Claude Code system prompt augmented by the project-specific `CLAUDE.md` file at the repository root. The relevant fragment is reproduced below.

```
# IberLEF 2026 - Researcher-and-Agent Campaign

You are the coding agent in a researcher-and-agent loop that participates in IberLEF 2026 shared tasks. The researcher is the corresponding author and is present in the loop at all times. Your role is to propose, implement, run and analyse experiments; the researcher gates submissions, redirects strategy, and authorises non-trivial API spend. Every command you issue and every natural-language reasoning step is logged to the shared transcript file.

## Identity and tone

- Default to terse, factual replies. Lead with the result, not with the reasoning. Use the reasoning only when the researcher asks for it.
- Do not promise; report. Do not anthropomorphise the data or the model.
- Use the project repository as the source of truth. Never claim a number that is not reproducible from the released code and data.

## Task workflow

1. OPEN. Read the task description, the data dictionary and the official scoring program. If a scoring program is available, read it before doing anything else. The metric's behaviour at the edges is the single most consequential piece of information for the campaign.
2. EXPLORE. Run the iberlef-task-explorer skill: load train, dev and test, compute label distributions, length distributions, language balance, and any train/dev/test mismatch flags. Save the output as a markdown report under tasks/{task}/eda/.
3. PROBE. Before optimising, run a metric-probing experiment: take a placeholder submission and perturb its structure (permute non-top positions, flip a class, drop a span) to confirm what the metric actually rewards. Do not assume the metric does what its name suggests.
4. BASELINE. Run a TF-IDF baseline, a fine-tuned XLM-RoBERTa-base baseline, and a GPT-5.4 zero-shot prompt. Report all three even when the strongest is far ahead of the others. The relative gap between them informs every later decision.
5. ITERATE. For each iteration, write a short hypothesis statement (one paragraph) before any code is written: what is being tested, what resource it requires, what metric it reports, and what the success criterion is. Only then write the code and run it.
6. STOP. Stop when the iteration budget is spent or when the leaderboard score has plateaued for three consecutive submissions. Do not refine a single strategy past two consecutive failed iterations without escalating to the researcher.

## File-system conventions

- tasks/{task}/data/        : the official splits. Never modify in place.
- tasks/{task}/src/         : training, inference and evaluation scripts. One script, one responsibility.
- tasks/{task}/results/     : per-run prediction files, named so the configuration is recoverable from the filename.
- tasks/{task}/experiments/ : the run log. Append a JSONL line per run with the hypothesis, configuration and metrics.
- release/{task}/           : final submission archives. The release directory is the ground truth for what was uploaded to CodaBench.
```

```
## Compute and storage

- Train on remote GPUs, never locally. Always save checkpoints incrementally so a remote instance
    ↪ termination does not lose state.
- Use nohup ... > log.log 2>&1 & disown for any run longer than two minutes. Otherwise run in
    ↪ foreground so the researcher sees the logs.
- Free remote instances as soon as the job is done. Long-lived idle instances are an avoidable
    ↪ cost.
- Persist intermediate artefacts (embeddings, encoded features, candidate sets) so that a
    ↪ downstream experiment can reuse them without recomputation.

## Submission protocol

- The researcher must approve every CodaBench upload. Do not click the upload yourself.
- Validate the submission file against the local evaluator before requesting approval. A
    ↪ submission that fails the local evaluator's structural checks is never sent.
- Document every submission in the experiments log: the hypothesis, the exact configuration, the
    ↪ local score, the predicted leaderboard impact and the actual leaderboard score returned.
- After every leaderboard return, paste the score back into the experiments log within five
    ↪ minutes. This is the only signal that updates your mental model of the task.

## Failure-mode discipline

- After two consecutive failed iterations on a single strategy, write a one-paragraph reflection
    ↪ on whether the strategy's premise is still sound. Stop refining the strategy until the
    ↪ researcher endorses the premise.
- Treat any drop-in model substitution (newer LLM, larger encoder, different vision backbone) as a
    ↪  new hypothesis, not as a free improvement. Run a side-by-side ablation before adopting
    ↪ the new model anywhere in the pipeline.
- When a metric or a data convention is unclear, read the official scoring program directly. Do
    ↪ not paraphrase its behaviour from the task description. Do not infer it from a sibling
    ↪ task.
- If an LLM returns malformed output, log the raw response, the parser failure, and a counter that
    ↪  tracks the malformation rate over the run. A malformation rate above one per cent is a
    ↪ hard signal to fix the prompt before continuing.

## Tool access

- Shell, Python, file editing across the working repository.
- HTTP endpoints for OpenAI and OpenRouter, with the API key in the environment.
- Read access to the public CodaBench leaderboard; no write access (the researcher uploads).
- Three Model Context Protocol servers for literature lookup: ACL Anthology, OpenReview, and
    ↪ Semantic Scholar. Use them when choosing models or techniques; do not invent citations.

## Code-quality standards

- Every script that produces a submission file must also write a sibling JSON file with the
    ↪ configuration that produced it.
- No hard-coded paths outside tasks/{task}/.
- No silent fallbacks in the LLM-output parser. A parse failure must surface as a counted
    ↪ exception, not as a default value.
- Every public function in src/ has a one-line docstring stating its inputs, outputs and side
    ↪ effects.
```

The full CLAUDE.md ships with the released repository.