Using RE-LLM Coding Uncertainty to Resolve Codebook Ambiguities: An Example of the CLARIFY Toolset and Workflow in Action

Fanjie Li[1,*], Madison Lee Mason[1], Daniel T. Levin[1] and Alyssa Friend Wise[1,*]

[1]*LIVE Learning Innovation Incubator, Vanderbilt University, TN, United States*

Abstract

Reasoning-Enhanced Large Language Models (RE-LLMs) enable new forms of human-AI collaboration in coding of student text that allows us to code better, not just faster. This paper introduces CLARIFY, a toolset and workflow to systematically elicit, quantify, and interpret LLM uncertainty during coding as a diagnostic resource for identifying and resolving codebook ambiguities. CLARIFY employs: (1) guided chain-of-thought for uncertainty flagging, (2) RE-LLM ensemble for committee-based uncertainty estimation, (3) consensus entropy calculation to prioritize high-uncertainty cases for expert review, and (4) a formalized process for iterative codebook and ground truth refinement. Initial application of CLARIFY is demonstrated through pilot analysis of nursing students' post-simulation reflections, tracing the active learning trajectory of a single code to show how it led to meaningful codebook clarifications and yielded improved performance on both training and held-out test sets.

Keywords

LLM-augmented content analytics, large language models, uncertainty quantification, human-in-the-loop, active learning, human-AI collaboration, hybrid-intelligence

1. Introduction

Over the past decade, a major direction within learning analytics and AIED research has been to automate coding of large volumes of student text to generate formative feedback for students, pedagogical insights for instructors, and adaptive support within personalized learning systems [1]. Yet many educational constructs that provide crucial insights into student learning are often complex, latent, and require more nuanced, conceptual understanding than traditional NLP approaches can readily attain [2]. Reasoning Enhanced Large Language Models (RE-LLMs), with their capabilities for conceptual inference and interpreting implicit meaning from unstructured text, now enable promising new approaches to this long-standing challenge. Moreover, unlike traditional classifiers that produce unexplained labels, RE-LLMs can articulate the logic chains leading to their coding decisions, enabling researchers to examine not just where disagreements arise but why. This explanatory capability enables new forms of human-AI collaboration in coding workflows, shifting AI coding tools from mere efficiency aids to collaborative partners, and from simply assisting humans to code *faster* to also augmenting human expertise to code *better*.

The promise of this shift is seen in an emerging body of LA and AIED research that explores novel applications of LLMs across different stages of the coding process. For example, LLMs can add a verification layer to human analysis, revealing when coders inadvertently deviate from code definitions across semantically similar examples [2]. LLMs' capacity to re-code the same data multiple times, without fatigue or influence of prior passes, can also be used to examine consistency across multiple runs [3]. Lack of stability in codes applied can point to where a codebook requires more careful op-

Joint Proceedings of LAK 2026 Workshops, co-located with the 16th International Conference on Learning Analytics and Knowledge (LAK 2026), Bergen, Norway, April 27 – May 1, 2026.

*Corresponding author.

fanjie.li@vanderbilt.edu (F. Li); madison.j.lee@vanderbilt.edu (M. L. Mason); daniel.t.levin@vanderbilt.edu (D. T. Levin); alyssa.wise@vanderbilt.edu (A. F. Wise)

0000-0001-7016-6354 (F. Li); 0000-0001-6395-0976 (M. L. Mason); 0000-0002-2652-0472 (D. T. Levin); 0000-0002-4043-6808 (A. F. Wise)

erationalization to achieve intra-coder consistency. The explanatory capacity of LLMs can also help researchers find codebook areas requiring further clarity and offer insights into how they might be revised to support more consistent application [2, 3]. Additionally, chain-of-thought reasoning can be combined with active learning to help diagnose faulty LLM logic in coding rationales, creating tight human-AI feedback loops that iteratively adjust model reasoning to improve alignment with human expert judgment [4].

Across these studies, an emergent insight is that coding disagreements, whether between human and AI coders, across multiple AI coders, or across multiple runs of the same AI coder, are not always just problems to be minimized, but can be mobilized as diagnostic signals for refining codebooks and identifying inconsistencies in initial human coding. This observation connects to work in both the learning sciences and machine learning, demonstrating how ambiguities that threaten coding consistency and accuracy can be identified through focused analysis of coding disagreements and difficulties. For example, researchers in the learning sciences have long recognized that examining the content and sources of inter-coder disagreements can be equally, if not more, valuable than achieving perfect agreement, as these disagreements often expose theoretically important cases that can help delineate construct boundaries and ambiguous code definitions [5]. In machine learning, the active learning paradigm similarly treats model uncertainty and decision difficulties as signals for identifying which examples are most informative for expert validation and model improvement [6]. Taken together, both perspectives recognize that decision difficulty, whether manifested as human inter-coder disagreement or model uncertainty, often signals latent ambiguity in the coding task, such as unclear construct boundaries that make borderline cases difficult to classify, or insufficient operational definitions that permit multiple plausible coding decisions.

While human-AI disagreement analysis has been increasingly utilized to refine codebooks and revisit codes applied [2, 3, 4], this has primarily been done retrospectively when coding conflicts that signal latent ambiguity happen to arise. However, with explicit prompting, RE-LLMs can now be instructed to proactively articulate uncertainty in their coding decisions as they navigate ambiguous cases, such as when a model settles on a code but with hesitation, recognizes competing interpretations, or identifies edge cases that require clearer coding guidance. This paper operationalizes this capacity through CLARIFY (**C**ollaborating with **L**LMs for **A**mbiguity **R**esolution via **I**terative **F**eedback on model uncertaint**Y**), a set of techniques, metrics, and a pipeline to systematically elicit, quantify, and interpret LLM uncertainty during coding as a diagnostic resource for improving codebook clarity and coding quality. Below we first situate CLARIFY within the evolution of human-machine collaborative coding approaches, and then present its core components and pipeline. Initial application of CLARIFY is demonstrated through pilot analysis of nursing students' post-simulation reflections, tracing the active learning trajectory of a single code to show how it lead to meaningful codebook clarifications and yielded improved performance on both training and held-out test sets.

2. Revisiting Approaches to Reconciling Human-Machine Coding Differences: From nCoder to LLMs-Assisted Content Analytics

Growing research on AI-assisted qualitative coding increasingly emphasizes designing effective human-AI collaboration to augment, rather than automate, human analysis [3]. Early approaches to automated coding have primarily focused on replicating human coding patterns across large datasets, training machines to reproduce human-labeled "ground truth," while issues of consistency and construct validity in those human labels often remain under-examined [7]. Yet, in practice, human coding frequently faces consistency challenges. For instance, coders may experience fatigue, drift from established definitions over time, and sometimes apply codes differently to similar examples as their working understanding of the category evolves across successive coding rounds [8]. These consistency challenges are often difficult to detect through traditional inter-rater reliability (IRR) measures alone, which assess coder agreement only on the double-coded subsets while leaving single-coded portions unverified. Moreover, while IRR effectively captures the level of agreement between coders, it

does not examine the reasoning or certainty behind their coding decisions. High IRR can still mask chance agreement or premature consensus based on weak or invalid coding rationales, leaving issues of construct validity largely unexamined.

Addressing these challenges requires approaches that examine not just agreement between coders but consistency and certainty in their coding logic across cases. Recent work has begun exploring how systematic analysis of human-AI coding disagreements can serve this diagnostic function, for instance, by detecting coder drift from codebook definitions, revealing competing interpretations of edge cases, or exposing ambiguities in construct operationalization and/or coding criteria [2, 3]. While early applications of human-AI disagreement analysis focused primarily on correcting machine errors, recent LLM-integrated workflows have expanded this scope to diagnose human coding inconsistencies and expose latent issues within the coding scheme [2, 3, 4]. Below we trace the evolution of these human-AI feedback mechanisms, examining how the role of disagreement analysis has expanded from calibrating machine coders to supporting humans' critical re-examination of coding decisions and construct clarity.

Disagreement analysis as a mechanism for driving iterative refinement of coding schemes is not a recent idea. In the pre-LLM era, while many researchers used traditional machine learning models for automated coding, nCoder [9] introduced a structured workflow that leveraged human-machine disagreement analysis to drive the iterative refinement of rule-based coding systems. Developed by Shaffer and colleagues in the late 2010s, nCoder is a platform that supports researchers in defining and validating keyword-based coding rule sets for automated data annotation. In nCoder's workflow, the coding scheme, operationalized as a set of regular expressions, is iteratively refined through cycles of human-machine coding comparison, disagreement analysis, and rule updates. The system formalizes an active learning cycle by first seeding an automated coder with human-defined wordlists and regular expression patterns, and then using reliability metrics (e.g., Cohen's Kappa) to monitor human-machine coding alignment on validation samples and flag discrepancies for researchers' closer examination. Through analyzing human-machine coding discrepancies, researchers can identify where the current coding rules fail to capture intended construct definitions, then update the regular expressions to reflect their refined understanding, creating a feedback cycle where human expertise continuously shapes machine coding behavior.

nCoder formalized two key mechanisms that are still central to human-AI collaborative coding approaches today. First, it operationalized disagreement as a diagnostic signal for refining coding rules, rather than solely as an indicator of system failure. Second, it demonstrated how refined human understanding could be fed back into the machine's coding logic through iterative feedback mechanisms. However, the limitations of NLP techniques available at the time imposed critical constraints on the system's ability to capture meaning beyond rigid keyword matches and the forms of human-AI collaboration nCoder could support. In particular, while the iterative refinement cycles in nCoder could effectively elicit human expertise to calibrate the automated coder, the machine's outputs provide limited basis for humans' critical examination of their own coding logic and assumptions. Establishing a bidirectional human-AI feedback loop, where the AI actively supports researcher reflection on their coding logic, requires capabilities beyond regular expression matching, such as the ability of AI coders to perform conceptual inference and explain the reasoning behind coding decisions. Recent advances in large language models have begun to provide these capabilities for machine coders, significantly expanding the nature and scope of human-AI interaction possible in this iterative feedback cycle. Below we examine three key new capabilities of (RE-)LLMs that enable richer forms of human-AI collaboration in qualitative coding, specifically: semantic understanding; explainable reasoning; and the capacity to sample diverse reasoning paths to assess self-consistency; and show how they support complementary human and AI roles either as a supplemental verification layer in the coding process or as collaborative partners for identifying and resolving gaps in a coding scheme (Figure 1).

First, leveraging LLMs' encoded semantic understanding, recent work has found that LLM coding can serve as a supplementary verification layer to detect drift in code application across entire datasets, identifying cases where comparable examples receive different labels and surfacing latent inconsistencies that traditional reliability metrics might miss. For instance, Zambrano et al. [2] demonstrated that,

Objectives	AI role	Human role
AI coding as a supplementary verification layer	Detect drift in code application across the entire dataset by leveraging LLM's encoded semantic structure to systematically reveal semantically similar examples that are coded inconsistently (Zambrano et al., 2023).	Resolve inconsistencies observed through human-in-the-loop validation (Cohn et al., 2024; Ramanathan et al., 2025; Zambrano et al., 2023).
	Assess intra-coder consistency through recursive model queries on the same example (Ramanathan et al., 2025; Tai et al., 2024), and use convergence of codes to quantify decision confidence (Wang et al., 2023).	Evaluate trends in code stabilization (Tai et al., 2024; Ramanathan et al., 2025) and review low-confidence coding decisions.
AI coder(s) as collaborative partners for identifying and resolving gaps in current coding scheme	Articulate reasoning behind its coding decisions that either prompts human reflection on their rationales and assumptions or exposes unclear code definitions that cause coder confusions (Ramanathan et al., 2025; Zambrano et al., 2023).	Perform disagreement analysis and resolve AI coder confusions either through codebook improvements or few-shot examples with human-corrected reasoning (Ramanathan et al., 2025; Zambrano et al., 2023; Cohn et al., 2024).
	Provide suggestions for updating the code definition upon human request (Zambrano et al., 2023).	Decide whether to accept/reject LLM's proposed changes (Zambrano et al., 2023).

Figure 1: Orchestrating human-AI collaboration while reconciling human-machine coding differences.

even after human coders established inter-rater reliability, LLM coding could help uncover remaining inconsistencies by identifying examples with similar meanings that were coded differently. This offers a scalable method for detecting coder drift and/or facilitating retroactive coding as human analysts' working understanding of the category deepens through repeated engagement with the data, enabling researchers to maintain consistency across large datasets without the burden of manual re-coding of earlier data.

Second, LLMs can articulate the intermediate thinking process behind their coding decisions through chain-of-thought prompting, a capability significantly enhanced in recent thinking and reasoning models. This extends disagreement analysis from simple label comparison to systematic comparison of coding rationales. When human and AI coders disagree, researchers can now probe the discrepancies in coding logic, examining whether disagreements stem from ambiguous construct boundaries, underspecified coding criteria, or competing interpretations of the code definitions or the data itself. This explanatory capacity has proven particularly valuable for exposing tacit assumptions in human coding logic. As demonstrated in recent studies [2, 3], LLM rationales that differ from human reasoning can help researchers to recognize where their own coding decisions rely on implicit criteria not captured in the codebook. Researchers can then address these ambiguities through codebook refinements and/or clarifying examples, creating a bidirectional feedback loop where both the analytic instrument (the codebook) and model performance improve through iterative refinement cycles.

Finally, leveraging LLMs' memoryless, stochastic properties, recent work has explored innovative use of recursive model queries to assess intra-coder consistency [3, 10]. Rather than relying on a single coding pass, researchers can run a model recursively on identical inputs, examining whether it arrives at stable coding decisions or exhibits decision variability [10, 11]. This approach draws on the self-consistency hypothesis [11], which posits that complex problems often admit multiple valid reasoning paths that converge on a unique correct answer, whereas flawed or less confident reasoning tends to be stochastic and divergent. In the context of data annotation, this means when coding criteria are clear and well-operationalized, different reasoning paths tend to converge on the same code, whereas ambiguous or underspecified definitions often produce inconsistent results across runs. High convergence across multiple passes thus signals decision stability and robust construct operationalization, while low self-consistency indicates models' uncertainty in the coding decision, which often reveals

edge cases or ambiguities in construct definitions that require human clarification [3, 11]. This enables researchers to identify not just where the model disagrees with humans, but where the model itself hesitates, a diagnostic signal we will explore further in subsequent sections.

The research trajectory traced above reveals a progression from detecting disagreements to orchestrating sophisticated human-AI collaboration while reconciling human-machine coding differences. However, current uses of human-AI disagreement analysis primarily operate downstream in a retrospective manner, intervening only after latent ambiguities have surfaced as explicit coding conflicts. Recent work has demonstrated that LLMs can express uncertainty during the coding process, for example through low self-consistency in recursive model queries [3, 10]. However, these uncertainty signals have been used primarily for post-hoc verification of coding quality, flagging potential issues in coded data rather than proactively guiding codebook refinement during the coding process.

CLARIFY (**C**ollaborating with **L**LMs for **A**mbiguity **R**esolution via **I**terative **F**eedback on model uncertaint**Y**) is a structured workflow that leverages model uncertainty as a proactive diagnostic resource. Rather than waiting for disagreements to emerge, CLARIFY surfaces cases where the model expresses uncertainty, for example where competing interpretations arise, where construct boundaries remain unclear, or where coding criteria prove difficult to apply consistently. This approach makes human-in-the-loop validation more efficient by directing expert attention to high-uncertainty cases that signal genuine need for human clarification, while using shifts in uncertainty patterns over time to empirically verify whether codebook refinements have successfully improved construct operationalization.

3. The CLARIFY Toolset and Workflow

CLARIFY operationalizes model uncertainty as a diagnostic resource through a structured pipeline that integrates uncertainty assessment, targeted disagreement analysis, and iterative codebook refinement and ground truth assessment (Figure 2). The workflow consists of three core stages: (1) uncertainty-aware coding with confidence quantification, (2) strategic case selection for human review, and (3) iterative codebook refinement via active learning. Below we detail each component and how they combine to create an efficient human-AI feedback cycle.

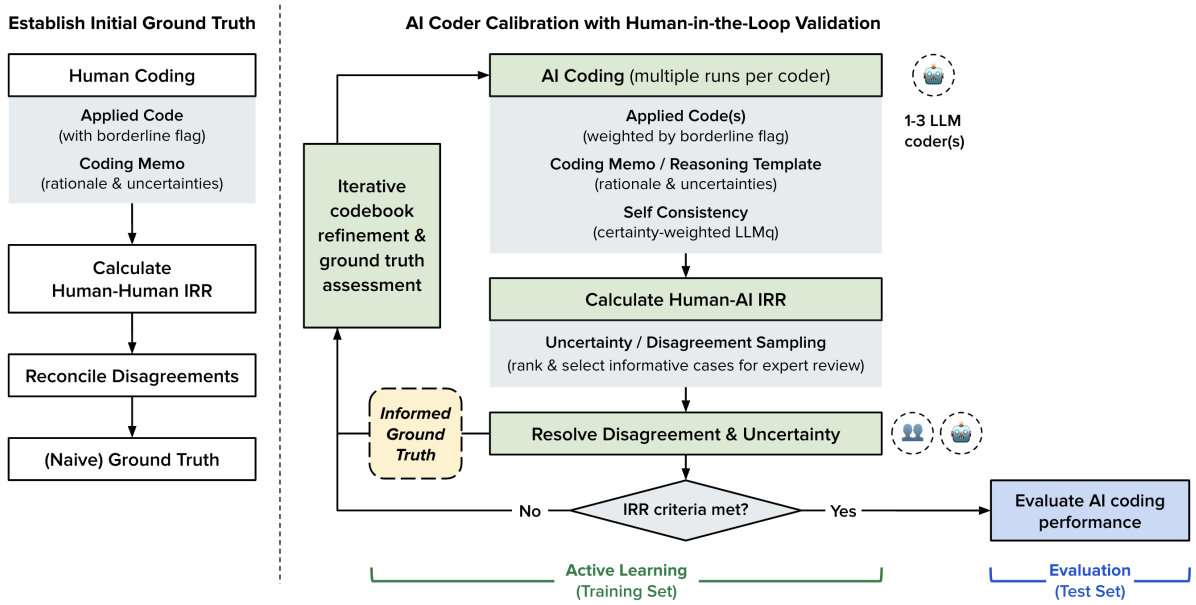

Figure 2: Key elements of CLARIFY toolset and workflow.

3.1. Stage 1: Uncertainty-Aware Coding and Confidence Quantification

Uncertainty assessment during coding: Guided chain-of-thought prompting is used to craft a reasoning template with two dedicated fields (***ambiguity assessment*** and ***uncertainty flag***), where the RE-LLM indicates when it finds code application difficult and articulates the source of uncertainty. Drawing on Ramanathan et al. [3], each AI coder is run multiple times on the same example to assess within-model consistency. This yields a ***certainty-weighted LLMq*** (Large Language Model quotient), accounting for both model self-consistency and certainty, computed by averaging coding outputs with uncertainty-flagged codes down-weighted.

Committee-based uncertainty estimation: Rather than relying on a single model's potentially biased (un)certainty estimate, multiple AI coders are employed, each using a different RE-LLM; their disagreement patterns provide more robust signals of genuine ambiguity. This ensemble approach, drawing on the ***Query by Committee (QBC) method*** from the active learning (AL) literature [12], gives us more robust estimates of uncertainty that are not subject to a single model's training biases or overconfidence.

3.2. Stage 2: Targeted Case Selection via Uncertainty Sampling

Using human-AI IRR monitoring to identify codes remain ambiguous: Computing IRR metrics between human and AI coders (e.g., Krippendorff's α [13]) reveals which specific codes remain challenging. Low IRR on a particular code signals that the codebook may lack clarity for that category or that the AI struggles with nuanced distinctions. The main goal of this diagnostic step is to identify which codes still need codebook refinement.

Use uncertainty sampling to prioritize the most "informative" cases warranting expert review or in-depth disagreement analysis: For each category with low IRR, rather than manually reviewing all disagreements, the CLARIFY toolset offers metrics to rank and select cases where the AI coder demonstrates the most uncertainty or where disagreements are most likely to expose meaningful boundary cases and codebook ambiguities. This active learning query strategy aims to make human-in-the-loop validation more efficient by prioritizing which cases need human expert attention. With a committee-based set up, selection of high-uncertainty cases that can inform codebook refinement is formalized using ***consensus entropy (CE)*** [14], a metric adapted from QBC which uses the Shannon entropy formula to combine within-model certainty and across-model consensus (ranges from 0 to 1, higher values indicate greater collective uncertainty) [15]. CE is used to systematically identify cases warranting human expert review, that can inform both code resolution and codebook improvement.

3.3. Stage 3: Iterative Codebook Refinement through Active Learning

Iterative codebook refinement and ground truth assessment: In each ***active learning*** round, the committee re-applies the updated codebook to the training set, using consensus entropy to surface cases that remain ambiguous. Human experts resolve LLM-flagged ambiguities, confirming initial codes or revising them based on the clarified codebook, aligning with Zambrano et al.'s [2] call for principled ground-truth revision via disagreement analysis. This cycle repeats for each code until reaching a pre-set stopping criterion (e.g., Krippendorff's $\alpha >= 0.75$). Between rounds, researchers examine how the committee's reasoning patterns and consensus entropy scores have shifted.

Validate on held-out test data once IRR criteria are met on training data: When a code achieves the target IRR on the training set, the final refined codebook is evaluated on the held-out test data. This validation step evaluates whether the AI coder generalizes well or has overfit to the training examples, revealing any remaining issues before deployment.

4. CLARIFY Toolset and Workflow in Action: An Example in Nursing Student Reflection Analytics

To demonstrate the practical utility of CLARIFY, we describe its use in pilot work with nursing students' post-simulation reflections. The analysis of such reflections is a domain where construct ambiguity can arise due to the complex, multifaceted nature of clinical reasoning and reflective practice. In the following sections, we trace the active learning trajectory of a single code to show how it led to meaningful codebook clarifications and yielded improved performance on both training and held-out test sets.

4.1. Context, Data and Coding Scheme

Reflect is a post-simulation reflection system designed to support nursing students' analysis of their simulation experiences [16]. It was developed as part of a larger project that leverages AI-powered tools to support experiential learning in nursing simulation. *Reflect* promotes sustained re-engagement with the simulation experience by having students segment first-person video into meaningful events and identify associated actions, goals, and cognitive states. The broader project's long-term goals include developing reflection analytics to support adaptive feedback, instructional insight, and indicators of competency development.

The pilot analysis uses an event-level reflection corpus from Vanderbilt University (Summer 2024) of 213 reflections from 16 students reflecting on key moments from a range of adult and pediatric simulations. The project's first coding scheme was a simulation-general activity framework aligned with models of clinical judgement [17] and self-regulation [18]. The codebook was comprised of 14 categories capturing what students reflect about (e.g., clinical reasoning, communication) and how they reflect (e.g., self-evaluation, challenging emotional experiences), developed iteratively through inductive analysis and deductive refinement grounded in nursing education and reflective learning theory [17, 18]. The resulting codebook prepared for use with AI coders included code definitions, indicators, and decision heuristics. Three researchers applied the codebook to double code 60 reflections (IRR: 0.63 < α < 0.91), with all disagreements reconciled. The example presented here focuses on the Interpreting Information to Validate Care Delivery Plan (VALIDATE) code.

To support AI coder calibration, the reconciled dataset was split into training (N = 36) and test sets (N = 24). When human coders initially disagreed, reconciled labels were entered as .3 / .7 (rather than 0 / 1) to indicate the uncertainty identified. Using the training data, three AI coders (o3, Claude-Sonnet-4, and DeepSeek-R1) were calibrated through iterative active learning using a shared system prompt comprising of: (1) a domain expert persona, (2) research context and goals, (3) guided chain-of-thought instructions, and (4) a dynamically-inserted and updatable codebook for each code. In each round of active learning, each LLM coder was run 10 times for a data instance; decisions flagged for uncertainty were similarly assigned .3 / .7 (rather than 0 / 1) across each LLM run. Average scores across runs were rolled up for each model to produce a certainty-weighted LLMq (LLMq-c). Committee-level uncertainty was indexed by consensus entropy (CE), computed by applying the Shannon entropy formula to the ensemble consensus score (i.e., ensemble-averaged vote share, LLMq-c) [14].

During each active learning round (see Section 4.2), overall agreement across the three models and human score for the code was evaluated against a target IRR threshold (Krippendorff's α >= 0.75). If agreement was below the IRR threshold, the six data instances with the highest-entropy (exhibiting the greatest committee-level uncertainty) were selected for targeted human review, leading to codebook refinement. This iterative process was repeated until the target IRR threshold on the training data was met. At this point active learning was complete and committee performance was evaluated on the held-out test data (N = 24) using both the initial and final codebooks to assess active learning performance gains.

4.2. Results

Figure 3 shows overall α for the VALIDATE code for each active learning round as well as the top 6 highest-entropy cases (out of 36 training examples), along with reconciled human code and each AI coders' LLMq-c.

AL round 1 (α = **0.641**, # disagreement = 12)

text id	CE	human	o3	claude	deepseek
213-12	0.99	1.00	0.79	0.09	0.30
216-3	0.99	0.70	0.94	0.31	0.27
206-13	0.94	0.70	1.00	0.31	0.56
214-5	0.94	0.70	1.00	0.57	0.30
217-10	0.93	0.00	0.77	0.44	0.18
206-12	0.93	0.70	1.00	0.63	0.30

AL round 2 (α = **0.739**, # disagreement = 9)

text id	CE	human	o3	claude	deepseek
216-3	0.97	0.30	0.97	0.69	0.42
212-21	0.94	0.00	0.97	0.03	0.45
201-12	0.93	0.00	1.00	0.13	0.25
210-12	0.92	0.00	0.97	0.16	0.21
206-12	0.91	0.70	1.00	0.63	0.30
214-5	0.87	0.70	1.00	0.68	0.46

AL round 3 (α = **0.845**, # disagreement = 5)

text id	CE	human	o3	claude	deepseek
216-3	1.00	0.30	0.88	0.65	0.30
206-13	0.96	0.70	0.94	0.53	0.30
201-12	0.82	0.00	0.82	0.00	0.21
202-14	0.81	0.70	1.00	0.90	0.41
201-11	0.80	0.70	1.00	0.94	0.38
206-12	0.80	0.30	0.06	0.31	0.30

Figure 3: Top 6 high-uncertainty cases based on consensus entropy (CE) in three active learning rounds.

As shown in Figure 3, as the codebook was iteratively refined across three active learning rounds, stronger agreement was achieved among the three AI coders and the human-reconciled code (α: 0.641 -> 0.739 -> 0.845). By round 3, the three AI coders and human score were in full agreement on all but 5 cases in the AL training set. In each round, human experts reviewed the top 6 high-uncertainty cases, selected by ranking training instances based on the committee's consensus entropy (CE). The AI coders' *ambiguity assessment* fields were used to surface competing rationales and diagnose sources of disagreement, which informed targeted refinements to code definitions and heuristics.

To illustrate this process, below we trace the codebook clarifications made for the VALIDATE code across three AL rounds. In Round 1, examination of the highest-entropy cases revealed AI coder uncertainty regarding how much detail a student should provide about their thinking process to satisfy the coding criteria for interpreting information to validate care delivery plan. For example, in reflection 213-12 ("…I was stressed with med calc…"), AI coders disagreed on whether mere reference to medication calculation activity constituted sufficient evidence of interpreting the information, resulting in near-maximal consensus entropy (CE=0.99; LLMq-c (o3) = 0.79, LLMq-c (claude) = 0.09; LLMq-c (deepseek) = 0.30).

Human adjudication traced this confusion to the absence of an explicit evidence threshold in the codebook. Because the project aimed to examine coding of what students were reflecting on rather than the depth or sophistication of the reflection, the research team established an inclusive decision rule and added the following explicit heuristic to the codebook:

> Apply this code when students are reflecting on interpreting information to validate care delivery plan, regardless of how they are reflecting on it. That is, reflection on interpreting information to validate care delivery plan can take various forms, including but not limited to describing what they did or failed to do during the simulation, evaluating their performance, expressing feelings or challenges about the experience, identifying lessons learned, and planning for future practice. All of these reflection types qualify as evidence for coding.

Following this clarification, the same reflection was coded with substantially lower uncertainty in Round 2 (CE=0.16; LLMq-c (o3) = 1.00, LLMq-c (claude) = 1.00; LLMq-c (deepseek) = 0.91)), with AI coding rationales explicitly citing the revised heuristics:

> **Sample rationale before codebook clarification:** "While 'med calc' nominally connects to an indicator, the lack of detail about validation behaviors creates uncertainty. Codebook requires evidence of cognitive work to validate care plans, not just acknowledgement that calculations occurred. However, one could argue the mere act of doing calculations implies validation attempt."

> **Sample rationale after codebook clarification:** "Minor uncertainty exists because the student does not explicitly describe the validation process (e.g., verifying dosage accuracy)

but only references stress associated with the task. However, the codebook's heuristics emphasize that outcomes or emotional reactions related to validation work still qualify as evidence."

High-uncertainty cases in subsequent rounds surfaced additional sources of ambiguity that prompted further codebook refinement. First, the wording of several code indicators proved insufficiently precise for AI coders, leading to over-generalized interpretations. For example, for the indicator "Confirming that the patient is properly prepared and positioned for the planned intervention," human coders readily distinguished between reflections describing verification thinking and those describing the execution of related procedures, drawing on their experience-based expertise. However, such distinctions proved problematic for the AI coders: at times assigning the VALIDATE code to reflections that described modifying nursing care (e.g., elevating a patient in response to family-reported dysphagia) without explicit evidence of interpretive validation work. To address this issue, we systematically tightened the codebook to more explicitly specify conceptual scope and exclude superficially related but non-verification-focused expressions in the reflections. Specifically, indicators were revised to require clear evidence of interpretive validation work, rather than mere execution of related hands-on procedures. For example, for the indicator "Verifying that specific environmental safety requirements are met before starting a procedure." "I maintained aseptic technique" would not qualify as evidence for VALIDATE whereas "I had to reglove to avoid contamination" or "I made sure the sterile field was still intact" would demonstrate validation thinking. Second, analysis of recurrent coder confusions revealed blurred boundaries with conceptually adjacent codes (e.g. Establishing Goals and Generating Solutions). To address this, we introduced explicit boundary-clarification heuristics to distinguish between them. For example, a heuristic was added to clarify the distinction between plan verification (i.e., confirming implementation details after a treatment decision has been made, which count as VALIDATE) and plan formulation (i.e., still deciding what treatment to provide, which does not count as VALIDATE), and supplemented these distinctions with few-shot examples demonstrating the intended coding logic.

To assess the effects of codebook refinement on unseen data, we applied the AI committee to a held-out evaluation set using both the initial and final codebooks. Results showed stronger agreement among the three AI coders and the human-reconciled code using the refined codebook compared to the initial version (α: 0.741 -> 0.941).

5. Conclusion

This paper introduces CLARIFY as a human–AI collaborative coding workflow that treats LLM uncertainty and disagreement not as failures to be minimized, but as diagnostic signals for improving codebook clarity. By combining uncertainty-aware LLM coding, committee-based disagreement analysis, and targeted human review through uncertainty sampling, CLARIFY supports experts in identifying latent ambiguities in the codebook and refining construct operationalization. Tracing the active learning trajectory of a single code, we demonstrate how high-uncertainty cases surfaced sources of ambiguity that were resolved by human coders through tacit expertise, yet remain challenging for LLMs. Codebook refinements driven by these LLM uncertainty signals led to stronger agreement not only on the training set, but also on a held-out evaluation set, indicating improved construct clarity rather than overfitting to training instances. Future work will extend this workflow to larger varied data sets, more complex coding schemes, and systematically examine the trade-offs among performance, human effort, and computational cost across different CLARIFY workflow configurations (e.g., committee size, number of LLM runs, and uncertainty aggregation strategies).

Acknowledgments

This work was supported by the National Science Foundation under Grant No. DRL-2418602. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Declaration on Generative AI

During the preparation of this work, the author(s) used GPT-5 to refine the text for grammar, spelling, and clarity of expression. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] V. Kovanovíc, S. Joksimovíc, D. Gašević, M. Hatala, G. Siemens, Content analytics: The definition, scope, and an overview of published research, in: C. Lang, G. Siemens, A. Wise, D. Gašević (Eds.), Handbook of Learning Analytics, 1st ed., SoLAR, 2017, pp. 77–92.

[2] A. F. Zambrano, X. Liu, A. Barany, R. S. Baker, J. Kim, N. Nasiar, From ncoder to chatgpt: From automated coding to refining human coding, in: ICQE2023, Springer, 2023, pp. 470–485.

[3] S. Ramanathan, L. A. Lim, N. R. Mottaghi, S. Buckingham Shum, When the prompt becomes the codebook: Grounded prompt engineering (GROPROE) and its application to belonging analytics, in: LAK'25 Proceedings, ACM, 2025, pp. 713–725.

[4] C. Cohn, N. Hutchins, T. Le, G. Biswas, A chain-of-thought prompting approach with llms for evaluating students' formative assessment responses in science, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2024, pp. 23182–23190.

[5] D. Hammer, L. K. Berland, Confusing claims for data: A critique of common practices for presenting qualitative research on learning, Journal of the Learning Sciences 23 (2014) 37–46.

[6] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, A. Fernández-Leal, Human-in-the-loop machine learning: a state of the art, Artificial Intelligence Review 56 (2023) 3005–3054.

[7] D. R. Thomas, C. Borchers, K. Koedinger, Beyond agreement: Rethinking ground truth in educational AI annotation, in: Proceedings of the Artificial Intelligence in Measurement and Education Conference, NCME, 2025, pp. 345–351.

[8] J. A. Holton, The coding process and its challenges, in: The Sage handbook of grounded theory, SAGE, 2007, pp. 265–289.

[9] Z. Cai, A. Siebert-Evenstone, B. Eagan, D. W. Shaffer, X. Hu, A. C. Graesser, ncoder+: A semantic tool for improving recall of ncoder coding, in: ICQE2019, Springer, 2019, pp. 41–54.

[10] R. H. Tai, L. R. Bentley, X. Xia, J. M. Sitt, S. C. Fankhauser, A. M. Chicas-Mosier, B. G. Monteith, An examination of the use of large language models to aid analysis of textual data, International Journal of Qualitative Methods 23 (2024) 16094069241231168.

[11] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-consistency improves chain of thought reasoning in language models, 2023. arXiv:2203.11171.

[12] R. Liere, P. Tadepalli, Active learning with committees for text categorization, in: Proceedings of the 14th National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence, AAAI'97/IAAI'97, AAAI Press, 1997, pp. 591–596.

[13] K. Krippendorff, Content Analysis: An Introduction to Its Methodology, SAGE, 2026.

[14] B. Settles, Active Learning Literature Survey, Technical Report TR1648, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[15] Y. Zhang, T. Liang, X. Huang, E. Cui, X. Guo, P. Chu, C. Li, R. Zhang, W. Wang, G. Liu, Consensus entropy: Harnessing multi-vlm agreement for self-verifying and self-improving ocr, 2025. arXiv:2504.11101.

[16] M. L. Mason, M. A. Jessee, D. T. Levin, Transforming experiences into expertise: Leveraging event cognition to support self-regulation in a practical learning system, 2026. In review.

[17] C. A. Tanner, Thinking like a nurse: A research-based model of clinical judgment in nursing, Journal of Nursing Education 45 (2006) 204–211.

[18] E. Panadero, A review of self-regulated learning: Six models and four directions for research, Frontiers in Psychology 8 (2017) 1–28.