A Computational Method for Measuring "Open Codes" in Qualitative Analysis

John Chen
University of Arizona, Tucson, AZ, USA
johnchen@arizona.edu
Alexandros Lotsos, Sihan Cheng, Lexie Zhao, Yanjia Zhang,
Jessica Hullman, Bruce L. Sherin, Uri J. Wilensky, and Michael S. Horn
Northwestern University, Evanston, IL, USA
{alexandroslotsos2026, sihancheng2026, xinyuezhao2020, fzhang}@u.northwestern.edu
{jhullman, bsherin, uri, michael-horn}@northwestern.edu

Abstract

Qualitative analysis is critical to understanding human datasets in many social science disciplines. A central method in this process is inductive coding, where researchers identify and interpret codes directly from the datasets themselves. Yet, this exploratory approach poses challenges for meeting methodological expectations (such as "depth" and "variation"), especially as researchers increasingly adopt Generative AI (GAI) for support. Ground-truth-based metrics are insufficient because they contradict the exploratory nature of inductive coding, while manual evaluation can be labor-intensive. This paper presents a theory-informed computational method for measuring inductive coding results from humans and GAI. Our method first merges individual codebooks using an LLM-enriched algorithm. It measures each coder's contribution against the merged result using four novel metrics: Coverage, Overlap, Novelty, and Divergence. Through two experiments on a human-coded online conversation dataset, we 1) reveal the merging algorithm's impact on metrics; 2) validate the metrics' stability and robustness across multiple runs and different LLMs; and 3) showcase the metrics' ability to diagnose coding issues, such as excessive or irrelevant (hallucinated) codes. Our work provides a reliable pathway for ensuring methodological rigor in human-AI qualitative analysis.

1 Introduction

Qualitative analysis is widely adopted across many social science disciplines. Most often, qualitative researchers apply descriptive labels (codes) in two ways: deductive coding, where codes are applied according to a preconceived coding scheme, and inductive coding ("open coding"), where codes are concepts derived from the raw data.

While methodologies such as Grounded Theory (GT) (Corbin and Strauss, 2008b, 1990) and Thematic Analysis (TA) (Braun and Clarke, 2006; Terry et al., 2017) rely on the inductive approach to discover emergent patterns from human data, inductive coding is hampered by its subjective and time-consuming nature (Attride-Stirling, 2001; Bowman et al., 2023). Since "ground truth" may not exist at this stage, truth-based evaluation methods (e.g., inter-coder reliability) mismatch inductive coding's inherent open-endedness (McDonald et al., 2019; Corbin and Strauss, 2008b; Terry et al., 2017). As many computational linguistics, machine learning, or qualitative research studies (e.g., Xiao et al.) attempt to leverage Generative AI (GAI) for inductive coding tasks, a theory-informed and computationally operational evaluation method is urgently needed.

This paper introduces a theory-informed computational method for systematically measuring open coding results from both human and machine coders. We propose an LLM-enhanced merging algorithm and four team-based evaluation metrics that do not rely on ground-truth assumptions. We conducted two empirical experiments to 1) reveal the merging algorithm's impact on the metrics; 2) validate the metrics' stability and robustness across multiple runs and different LLMs; and 3) showcase the metrics' ability to diagnose coding issues, such as excessive or irrelevant (hallucinated) codes. The results illustrate the metrics' stability, robustness, and use cases, with caution notes regarding their limitations and potential risks.

2 Related Work

2.1 The Nature and Challenges of Inductive Qualitative Coding

During inductive coding, researchers identify concepts and themes directly from raw data (Strauss and Corbin, 1998; Rahman, 2016), aiming at discovering novel insights often without an existing theoretical framework (Corbin and Strauss, 2008b, 1990; Strauss and Corbin, 1998; Terry et al., 2017).

Findings of the Association for Computational Linguistics: ACL 2026, pages 41740–41758

The process involves iterating through a corpus to identify meaningful segments and assign descriptive labels (codes) that emerge directly from the data. Researchers then often group their coded segments into labeled "categories" that are used for further analysis. This process resembles iterative clustering in machine learning contexts. Inductive coding is open-ended, subjective, and does not strive for a singular "correct" result (Terry et al., 2017). Rather, the process should capture as many aspects, patterns, or "codable moments" as possible (Terry et al., 2017; Corbin and Strauss, 1990, 2008a).

Yet, inductive coding is inherently subjective, time-consuming, and prone to ambiguities, making methodological rigor difficult to achieve (Attride-Stirling, 2001; Bowman et al., 2023; Braun and Clarke, 2021; Bringer et al., 2004; Tuckett, 2005; Furniss et al., 2011; Saunders et al., 2018). Since the process aims to widely capture novel insights rather than enforcing consistency, deductive coding metrics (such as inter-rater reliability) become gravely inadequate due to their reliance on "ground truth" assumptions (McDonald et al., 2019).

To address this mismatch, qualitative researchers are shifting towards team-based approaches, where the team constantly compares and contrasts codes from multiple individuals (Cascio et al., 2019; Thomas, 2006). Team-based approaches embrace different perspectives, resulting in more insights and mitigating individual biases (Corbin and Strauss, 1990; Thomas, 2006). It makes researchers closer towards the elusive goals of inductive analysis, such as depth, variation, and theoretical saturation (Corbin and Strauss, 2008b; Adams et al., 2008; Furniss et al., 2011; Saunders et al., 2018).

2.2 Evaluating ML/GAI for Inductive Qualitative Coding

ML/GAI approaches offer significant potential to support and enhance qualitative research by assisting in the coding process (Xiao et al., 2023). Existing computational approaches have primarily framed machine-assisted qualitative coding as either a classification-based task, which mimics human labels, or a generation-based task, which produces codes directly from data (Liew et al., 2014; Gebreegziabher et al., 2023; Rietz and Maedche, 2021; Xiao et al., 2023; Grootendorst, 2022; Saravani et al., 2023; Sievert and Shirley, 2014; De Paoli, 2023a; Sinha et al., 2024).

Effectively leveraging ML/GAI's potential requires robust evaluation methods that account for the open-ended and exploratory characteristics of inductive coding. However, existing evaluation strategies remain largely misaligned with these goals:

1. **"Ground truth"-based metrics** compare an input set of codes against an expert-labeled dataset (Parfenova et al., 2025; Zhao et al., 2024; Dai et al., 2023). While it provides quantifiable metrics such as precision and recall, its presupposition of a single correct answer directly contradicts qualitative research theories (Corbin and Strauss, 2008b; Terry et al., 2017). Essentially, this approach constrains inputs to a predefined scope, thereby limiting the discovery of novel insights (Liew et al., 2014; Xiao et al., 2023; Parfenova et al., 2025).

2. **Clustering and topic-coherence metrics** evaluate internal structure through measures such as compactness, separation, or word-level coherence (Rahimi et al., 2023). Although they do not require ground truth, they prioritize internal homogeneity rather than conceptual breadth or interpretive variation. Inductive coding, by contrast, values the coverage of diverse and crosscutting ideas rather than tight cluster cohesion (Corbin and Strauss, 2008b; Terry et al., 2017). Moreover, these metrics treat each coder or model output in isolation, lacking a mechanism to assess how different coders align, diverge, or collectively represent the dataset (Thomas, 2006). Hence, their objectives are incompatible with the theoretical aims of inductive qualitative analysis.

3. **Human-annotated evaluations** ask experts to assess the usefulness, explainability, or relevance of machine-generated codes through survey-based instruments (De Paoli, 2023a,b; Zambrano et al., 2023; Spinoso-Di Piano, 2023). While such methods provide valuable qualitative insights, they are costly, time-consuming, and difficult to scale for large datasets or iterative evaluation cycles, potentially overlooking systematic omissions when all coders fail to recognize a key concept (Parfenova et al., 2025).

3 Computational Metrics

Building on the team-based approach adopted by qualitative researchers (Thomas, 2006), our method aggregates multiple coders' coding results (i.e., codebooks) into a shared analytical space and calculates four computational metrics. Guided by qualitative analysis theory and the limitations identified in existing evaluation methods, our metric design follows three core requirements: First, the evaluation must remain independent of any ground truth so that coders can be assessed even when no authoritative codebook exists. Second, it must capture both the breadth of ideas and the interpretive balance of each coder while discouraging over-fragmentation or redundant labeling. Third, it must be stable across different models and repeated runs while remaining sensitive to problematic cases such as hallucination or flooding.

To address these goals, we introduce four complementary metrics: **Coverage**, **Overlap**, **Novelty**, and **Divergence**. Together, they evaluate how individual coders contribute to the collective interpretation of the data. Our method operates without assuming a ground truth or requiring human adjudication, though expert-coded results can serve as useful anchors when evaluating untested models or prompts. A reference implementation of the aggregation algorithm, full documentation, and metric calculation procedures are publicly available (open-sourced software package, licensed under CC BY–NC 4.0).

3.1 Aggregating Coding Results: Code Spaces (CSP) and Aggregated Code Spaces (ACS)

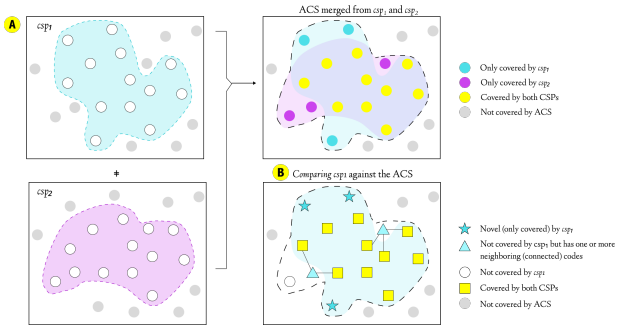

Figure 1: **A**: A conceptual illustration of an ACS merged from csp_1 and csp_2. **B**: Measuring csp_1 using the merged ACS as a reference.

The first step towards calculating our proposed metrics is to aggregate the codebooks produced by multiple individual coders into a single conceptual space that serves as an approximation of "all possible interpretations" of the data, as prescribed by qualitative analysis methods (Fig. 1).

To do this, we first consider each coder's results and define their **Code Space** (CSP), to be the set of all codes they identified or interpreted from the dataset (Fig. 1A). In turn, we can consider the union of all individual CSPs as the **Aggregate Code Space** (ACS) that encompasses the codes identified or interpreted from the dataset by all coders. However, simply considering the ACS to be this union does not take into account that real-world coders often use different codes to represent identical or very similar ideas (e.g., the codes "User Feedback" vs "Feedback from User").

To account for this, we propose a four-stage algorithm that utilizes semantic similarity and hierarchical clustering to iteratively consolidate an ACS from a set of individual CSPs. In addition to this, we also expand our definition of an ACS to include the fact that semantically similar codes are connected by links as $Neighbors$ (Fig. 1B). Noting that each code in an individual CSP has a $label$ (the code itself) and *may* have $examples$ (pieces of data the code was applied to), and a $definition$ (a description of the concept the code covers), the algorithm is thus defined as follows:

1. Begin with the ACS simply being the union of all individual CSPs, only considering the $label$ of each code.

2. For each code in the ACS, use hierarchical clustering to merge codes with semantically very similar $labels$. For each merging pair, the shorter label will be adopted.

3. For each code in the ACS, use LLMs to generate a new $definition$ based on its $label$ and $examples$. Then, repeat the merging using both $label$ and $definition$, using LLMs to generate the resulting $label$ and $definition$ of the merged code.

4. Finally, repeat stage 3 but iteratively and using our modified clustering algorithm (see below for more details).

In stages 2 and 3, we apply a strict threshold for cosine distances between the text embeddings of each code. However, we found that a single threshold was often insufficient in separating different codes. In stage 4, we therefore adapt hierarchical

clustering to apply two merge thresholds ($lower$ and $upper$, with $lower < upper$) on each dendrogram node, from which the $penalty$ coefficient is calculated. These penalty terms prevent merges that would conflate distinct concepts and ensure that smaller or sparser codebooks do not exert disproportionate influence in the merged structure.

Algorithm 1 First Penalty on the Difference Between Examples

Require: Candidate codes $A, B \in ACS$ with embeddings c_a, c_b and example sets E_A, E_B

$e = \frac{|E_A \cap E_B|}{|E_A \cup E_B|}$ ▷ e: example-overlap ratio (Jaccard)

$dist_{a,b} = d(c_a, c_b) + penalty * e^2$ ▷ d: cosine distance

Algorithm 2 Second Penalty on Unique Examples of the Potential Merge

Require: $dist$ = adjusted distance of the node

Require: $count$ unique examples of the node

Require: $count_{avg}, count_{max}$: average and maximum unique-example counts across candidate nodes

$o = max(\frac{count - count_{avg}}{count_{max} - count_{avg}}, 0)$

if $dist \leq lower$ **then**
    return YES
else if $dist > upper$ **then**
    return NO
else if $dist + penalty * o^2 < upper$ **then**
    return YES
else
    return NO ▷ Considered as "Neighbors"
end if

3.2 Four Computational Metrics

The four proposed metrics measure each coder's CSP against the ACS in four ways: Coverage, Overlap, Novelty, and Divergence (Fig. 1B). In Algorithms 3–6, x denotes a coder, c a code in ACS, $obs_{x,c}$ the observation score, and $score_c$ the aggregate code weight.

- **Coverage**: How much conceptual space does a given CSP cover in the ACS? Both TA and GT strive for "richness" of codes, capturing depth and variation for further analysis (Corbin and Strauss, 2008b; Braun and Clarke, 2013). Each code is weighted by the number of coders who identified it, and each coder is weighted by their total number of codes to prevent inflation from over-coding. Coverage increases as a coder contributes more distinct codes that expand the shared conceptual space, reflecting broader interpretive reach. Excessively high values, particularly when accompanied by many codes, may indicate redundancy or flooding.

Algorithm 3 Weighting Codes and Codebooks

for $x \in ACS$ **do**
    $size_x = \max\{\#(code \in x), size_{median}\}$
    $weight_x = \frac{1}{\ln(size_x)}$
    for $code \in ACS$ **do**
        if $c \in x$ **then**
            $obs_{x,c}$ *= 1
        else
            $obs_{x,c}$ *= $\frac{\ln(|neighbors \in x| + 1)}{\ln(|neighbors| + 1)}$
        end if
        $score_c$ += $obs_{x,c} * weight_x$
    end for
end for

Algorithm 4 Calculating Coverage

for $x \in ACS$ **do**
    $coverage_x = \frac{\sum_{c \in acs} obs_{x,c} \times score_c}{\sum_{c \in acs} score_c}$
end for

- **Overlap**: How much does a given CSP overlap with others? Overlap increases as a coder's codes coincide or neighbor those contributed by others, indicating stronger conceptual alignment and agreement. Very high Overlap combined with low Novelty may suggest redundancy, whereas very low Overlap can indicate conceptual drift, inconsistency, or misunderstanding. The algorithm is almost the same as the one proposed for coverage, except each coder's impact on $score$ is removed from the ACS (see Divergence).

- **Novelty**: How many "unique" codes does a given CSP include (i.e., codes that no other coders identified)? Novelty increases when a coder introduces previously unseen ideas, capturing their contribution of new conceptual ground. Moderate Novelty indicates balanced and productive insight, while excessive Novelty with low Overlap may suggest idiosyncratic or hallucinatory coding. In considering

Novelty, we measure how much additional conceptual space a coder contributes beyond the team's collective coverage.

Algorithm 5 Calculating Novelty

for $x \in ACS$ **do**
    $novelty_x = \frac{\sum_{c \in csp(novel=1)} obs_{x,c} * score_c}{\sum_{c \in acs(novel=1)} score_c}$ ▷ $novel = 1$: codes not identified by other coders
end for

- **Divergence**: How far is a given CSP's code distribution from the ACS? Divergence increases as a coder's focus or emphasis departs from the group's shared distribution, signaling either distinctive analytical perspective or instability. Lower Divergence indicates stronger alignment with the team baseline. We calculate each CSP's divergence as the separation from its probability distribution from that of other CSPs. We used the Jensen-Shannon Divergence (JSD) to tolerate potential zeros.

Algorithm 6 Calculating Divergence

for $x \in ACS$ **do**
    $B_c = score_c - obs_{x,c} * weight_x$ ▷ Baseline - excluding the coder's contribution
    $divergence_x = \sqrt{JSD(B \parallel obs_{x,c})}$
end for

Because coding traditions vary, these metrics are best interpreted in combination rather than through fixed thresholds. In practice, productive coders tend to maintain moderate Coverage together with Overlap that reflects shared understanding and Novelty that contributes distinct yet relevant ideas. Exact ranges will vary by dataset and analytical context.

4 Experimental Design

To empirically validate our computational metrics and merging algorithm, we designed two experiments to answer three research questions (RQ1-RQ3):

1. How does each stage of our merging algorithm affect our metrics?
2. Given the probabilistic nature and different capabilities of LLMs, how robust or stable are our metrics?
3. Can our metrics identify edge cases such as excessive codes or hallucinations?

4.1 Task and Dataset

We reused the prompt, dataset, and human coding results from Chen et al.'s study, where researchers conducted manual evaluation. In each experiment, we applied our computational metrics to inductive qualitative coding results from four human coders (three PhD students, one undergraduate student, all in a U.S. higher education institution) and four machine coders (the same LLM with different prompts). Both experiments work on an online conversation dataset between Physics Lab (an online learning software)'s designers and teacher users. The conversation happened in public messaging groups, and the collection of such data has been approved by a university IRB. The dataset is attached as part of the Software package and has been properly anonymized. Our usage of the dataset is consistent with the original intention and IRB approval.

Due to human researchers' limited capacity, we focused on the first 127 messages. The same question was provided to human and machine coders: "How did Physics Lab's online community emerge?"

Since most qualitative data are from human subjects and are subject to IRB protection, we intentionally chose open-source and locally available models for the experiments. We used Gemma3-27B (with temperature = 0.5) to generate new sets of machine codes. We used mxbai-embed-large to calculate semantic distances (Lee et al., 2024; Li and Li, 2023).

4.2 Experiment 1: Ablation and Comparison Study

Experiment 1 addresses RQ1 and RQ2 through an ablation study on the four stages of our merging algorithm . We chose four machine coders from Chen et al.'s comparison study: Chunk-Level (i.e., generate per "chunk" of messages); Chunk-Level, Structured; Item-Level (i.e., generate per message); Item-Level, Verb Phrases Only.

1. **Condition 1** corresponds to the first "naive" stage, where codes are merged solely by their labels.
2. **Condition 2** corresponds to the second stage, where codes are merged with a strict threshold (0.32) by their labels.

3. **Condition 3** corresponds to the third stage, where an LLM generates definitions based on each code's label and examples. Then, the codes are merged with a strict threshold (0.32) by labels and definitions.

4. **Condition 4** corresponds to the fourth stage, where the codes are iteratively merged with an upper threshold (0.55) and a lower threshold (0.32) by labels and definitions, until no more codes can be merged.

For each condition and LLM used, we repeated 10 runs, recorded each human and machine coder's number of codes within the merged ACS, and calculated four computational metrics. In addition to individual coders, we also calculated metrics for the combination "group" of AI or human coders. Thresholds in each condition are chosen interactively through our example implementation. The strict threshold is chosen by ensuring that 10 code pairs with a semantic distance right below it have the same meanings. The upper threshold is chosen by ensuring that 10 code pairs right below it have at least similar meanings.

In Stages 3 and 4, the study doubles as a comparison between different LLMs used in the process: Gemma3 27B (non-reasoning, small, open-source) (Team, 2025a), Qwen QwQ 32B (reasoning, small, open-source) (Team, 2025b), GPT-4.1 (non-reasoning, large, proprietary), and Gemini-2.5-Pro (reasoning, large, proprietary).

4.3 Experiment 2: Measuring Edge Cases

Experiment 2 addresses RQ3. Starting from the Item-Level coder, which performed the best (together with its Verb Phrases Only Variant) in human evaluation and computational metrics, we created three variants to simulate potential edge cases:

1. **Flooding Coder** is explicitly instructed to generate an excessive number of codes per item.

2. **Hallucinating Coder** has the same prompt but works with an irrelevant, AI-generated conversation.

3. **Hallucinate + Flooding Coder** combines the two changes together.

For each variant, we repeated 10 runs with the same human coders and machine coders, replacing results from the Item-Level coder with its variant. Since our preliminary results find little impact on LLM choice, we only used Gemma3 27B for this experiment. In total, both experiments cost $\tilde{8}$ million LLM tokens (5 read, 3 write), around $20 for proprietary models.

5 Empirical Study

The following sections present our hypotheses and empirical results. We provide more details through Appendices B and via the reproduction repository.

5.1 RQ1: Measuring Each Stage's Impact on Our Merging Algorithm

Experiment 1 first examines how each stage of our merging algorithm impacts each coder's number of merged codes and the resulting metrics.

5.1.1 Hypothesis 1: Evaluation condition significantly affects the number of merged codes and computed metrics.

We used ordinary least squares (OLS) regression to model the effect of algorithmic stages (Conditions 1-4) on the number of consolidated codes and each coder's four metrics.

Result: **Mostly Confirmed**. Each algorithm stage significantly reduced the total number of merged codes ($p < 0.001$). As shown in Table 1, we observed significant shifts in computational metrics in Conditions 3 and 4, but not in 2.

Condition	Coverage	Overlap	Novelty	Divergence
Condition 2	0.09%	-0.09%	0.05%	0.37%
Condition 3	3.60%	5.45%	0.94%	-4.31%
Condition 4	7.02%	7.86%	-1.64%	-1.91%

Table 1: OLS regression coefficients for evaluation metrics across Conditions 2 to 4 (relative to Condition 1). Conditions 1 and 2 are deterministic. For other values, p < 0.001.

5.1.2 Hypothesis 2: Evaluation condition has minimal impact on the relative ranking of coder metrics.

We conducted a ranking stability analysis across conditions using one-way ANOVA with Tukey HSD post-hoc comparisons.

Result: **Partially Confirmed**. While algorithmic stages shift the values of computational metrics, rankings remain relatively stable. For all metrics, rankings of top performers (#1-5) stay the same. For other coders, rankings within the label-only (1, 2) and LLM-enriched conditions (3, 4) are highly similar.

5.2 RQ2: Evaluating the Robustness and Stability of Our Proposed Metrics

Experiment 1 then evaluates whether our computational metrics remain robust across repeated runs and different LLMs in Conditions 3 and 4.

5.2.1 Hypothesis 3: LLM used in the merging process significantly influences metrics and code counts.

We used OLS regression to model the effect of LLMs on the number of consolidated codes and each coder's four metrics, controlling for fixed effects between Conditions 3 and 4 and between individual coders.

Result: **Partially Confirmed.** Across Conditions 3 and 4, three models (Gemma3 27B, Qwen QwQ 32B, and GPT 4.1) produce very similar metrics and numbers of merged codes. The only substantial deviation comes from *Gemini-2.5-pro*, which produces fewer merged codes, higher coverage and overlap (approximately 4% to 6% increase), and lower divergence (4% decrease) (Fig. 2).

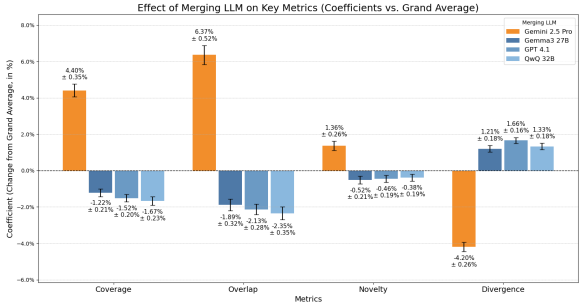

Figure 2: Effect of merging LLM on four evaluation metrics from the OLS model. Each bar represents the average coefficient difference from the grand mean across LLMs, along with 95% confidence intervals.

5.2.2 Hypothesis 4: Metric outcomes are well-explained by condition, model, and coder identity.

From the same OLS model used by Hypothesis 3, we calculated adjusted R^2 values to determine the extent to which the combination of condition, merging LLM, and coder explains variation in metric values.

Result: **Confirmed**. All adjusted R^2 values exceed 0.91.

5.2.3 Hypothesis 5: Repeated measurements under the same condition/model yield low coefficients of variation (CoV).

We calculated the coefficient of variation for each metric over 10 evaluation runs per Condition per merging LLM.

Result: **Confirmed**. CoV values remain below 0.1 in all cases. Divergence has the lowest variability around 0.01. Condition 4 shows slightly higher variance than Condition 3.

5.2.4 Hypothesis 6: LLMs used in the merging process have little effect on the relative ranking of coders.

We conducted one-way ANOVA with Tukey HSD post-hoc comparisons to examine whether coder rankings differed across LLMs.

Result: **Mostly Confirmed**. The top and bottom-ranked coders remained consistent across all four LLMs in both Conditions 3 and 4. Rankings for mid-performing coders fluctuate, primarily when their differences are not statistically significant.

5.3 RQ3: Testing Our Proposed Metrics' Diagnostic Utility for Edge Cases

Experiment 2 tests whether our computational metrics can detect abnormal inductive qualitative codes from machine coder variants designed to simulate edge cases: Flooding, Hallucinating, and Combined (Fig. 3).

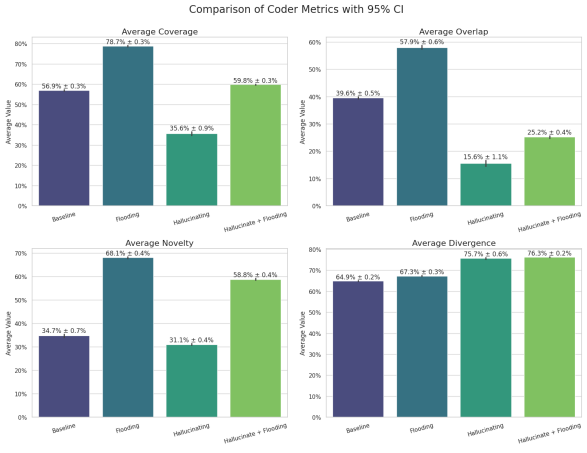

Figure 3: Mean coder metrics across Baseline, Flooding, Hallucinating, and Combined variants with 95% confidence intervals.

5.3.1 Hypothesis 7: Excessive coding increases coverage, overlap, and novelty with diminishing returns, while divergence remains stable.

Result: **Confirmed**. The mean metrics from the flooding coder have higher coverage (78.7%), overlap (57.9%), and novelty (68.1%) than the Baseline coder. Novelty showed diminishing returns. Divergence remained stable (67.3% vs. 64.9%). A similar effect is observed between the Hallucinating and Combined coders.

5.3.2 Hypothesis 8: Coding on irrelevant data (i.e., hallucination) reduces coverage and overlap, while increasing divergence.

Result: **Confirmed**. The mean metrics from the Hallucinating coder have reduced coverage (35.6%) and overlap (15.6%), while divergence increases sharply to 75.7%. A stronger effect is observed between the Hallucinating and Combined coders. In particular, while the Combined coder produced 245% more codes than the baseline (1,775 vs. 514), it has lower overlap (39.6% vs. 25.2%) and higher divergence (76.3% vs. 64.9%).

6 Discussions

This paper introduces a theory-informed computational method for systematically evaluating the outputs of inductive coding. At the heart of our contribution are four metrics designed to capture the multifaceted nature of inductive coding:

- **Coverage** measures the depth and variation of a coder's contribution against the Aggregated Code Space (ACS), reflecting the qualitative goal of achieving breadth and depth in analysis (Corbin and Strauss, 2008b).
- **Overlap** quantifies a coder's alignment with the conceptual consensus of the group, indicating how much their interpretations resonate with others.
- **Novelty** identifies the unique concepts a coder introduces, highlighting their potential value in bringing new perspectives to the analytical process.
- **Divergence** measures how much a coder's conceptual focus differs from others in the group, offering insight into their unique analytical lens.

Taken together, these metrics provide a holistic and nuanced assessment of a coder's performance that does not rely on "ground truths" that may not exist even with teams of human experts. Instead of pursuing a single, simplistic measure of agreement, our method embraces the subjectivity and exploratory spirit of inductive qualitative analysis (Corbin and Strauss, 1990; Braun and Clarke, 2012). It enables researchers to appreciate not only consensus but also the valuable variation that different coders bring to the table, providing a more complete picture of human-AI or human-human collaboration processes.

6.1 The Necessity of the Iterative, LLM-Enriched Merging Algorithm

RQ1 explored the impact of each stage of our merging algorithm. The findings from our ablation study (see 5.1) show that while each stage progressively reduces the number of codes in the Aggregated Code Space (ACS), the most significant shift in our metrics occurs between Stage 2 and Stage 3. This transition, which introduces LLM-generated definitions to the merging process, is far more impactful than simply relaxing semantic distance thresholds, as seen in the transitions between other stages.

This result confirms a central premise of our work: capturing true conceptual similarity in inductive coding requires moving beyond superficial linguistic parallels. The "naive" merging stages (Conditions 1 and 2), which rely solely on code labels, are insufficient. Therefore, the introduction of LLM-generated definitions centrally contributes to our proposed metrics. This stage infuses the semantic representation of each code with meaning derived from its underlying data points ("examples"), without forcing a direct comparison between those data points. Such insulation is vital for the inductive workflow, where coders may legitimately identify the same concept in different segments of the data (Corbin and Strauss, 1990); comparing data points directly would be prone to error.

Hypotheses 1 and 2 also clarify the distinct roles of Stage 4 (iteratively merging with more relaxed thresholds), which produces a highly consolidated ACS but does not significantly impact the measuring outcome. Since we regenerate a label and definition for each merging pair, Stage 4 is computationally intensive. Therefore, its utility depends on the research goal. To produce a clean conceptual map for further interpretation, comparison, or synthesis (e.g., when identifying a coder's poten-

tial bias), the computational cost of Stage 4 is well justified. Conversely, when numerical metrics are the primary output (such as in large-scale model comparisons), the less computationally demanding Stage 3 may provide sufficiently stable and reliable outcomes for screening purposes.

6.2 The Impact of LLM on Outcomes

While the involvement of LLMs in the merging processes introduces intrinsic randomness, the findings from RQ2 confirm our method's reliability and robustness across multiple LLMs and repeated runs. As shown in Hypothesis 4, with very high adjusted R^2 values, our proposed metrics are overwhelmingly explained by coders' intrinsic merits. Repeated measurements under the same conditions yield low coefficients of variation (Hypothesis 5), showing stability across multiple LLM samplings.

Crucially, our metrics' stability extends to the choice of LLM. Tested on a diverse set of models, the choice of model has little effect on the relative ranking of coders (Hypothesis 6), where the top and bottom-ranked performers remained consistent across all LLMs. This implication is significant for the practical adoption by qualitative researchers, as they can confidently use smaller, open-source models (e.g., Gemma3 27B) to measure outcomes from human or machine coders. Such models can be easily deployed locally, providing better privacy protections and regulation compliance for handling often sensitive human subject data.

6.3 Performance of Metrics in Edge Cases

RQ3 evaluated the robustness and diagnostic utility of our metrics in simulated edge cases. The "Flooding" coder variant, explicitly prompted to produce an excessive number of codes, was expected to create significant redundancy. The metrics correctly capture that: both "Flooding" variants registered higher Coverage and Novelty, but with diminishing returns that mark the results' redundancy.

Similarly, the "Hallucinating" coder variant, which worked from irrelevant data with the correct prompt, was expected to produce thematically similar codes but fail to capture critical details from the true data. The metrics capture the issue as well: both "Hallucinating" variants produced a predictable and dramatic drop in Coverage and Overlap, coupled with a sharp increase in Divergence.

In all three cases, our metrics behave predictably in response to those coding issues, suggesting that abnormal comparative values from our metrics can serve as a "red flag" for further evaluation. This diagnostic capability offers a layer of quality control that is essential for ensuring rigor in human-AI collaborative workflows.

7 Conclusion

This paper presents a reliable and robust computational method for the theory-informed, systematic evaluation of inductive coding. By moving beyond a reliance on a single "ground truth," our approach provides a practical pathway for qualitative researchers to leverage AI in inductive analytical processes responsibly. By measuring and quantifying conceptual ideas like coverage, overlap, novelty, and divergence, we shift the evaluation focus of inductive coding from enforcing agreement to a more nuanced appreciation of the diverse contributions that each coder brings to the collaborative, exploratory process. As more and more qualitative researchers explore LLMs for inductive coding, we offer a timely contribution to ensure methodological rigor and facilitate more effective and transparent human-AI collaboration.

8 Limitations

While this study presents a robust computational method for evaluating inductive coding, it is important to acknowledge its limitations. From there, future research can help establish best practices and clear guidelines for qualitative researchers to adopt our metrics. Application and misuse risks are discussed separately in Appendix A.

8.1 Limitation on the Dataset and Domain

The validation of our metrics was conducted on a single dataset consisting of online conversations from a specific online community. While effective in this context, the method's performance and the metrics' utility need to be tested on more diverse forms of qualitative data, such as semi-structured interview transcripts or open-ended survey responses. The nuances of these different data types may present unique challenges not encountered in this study.

8.2 Limitation on the Simulated Edge Cases

The "Flooding" and "Hallucinating" coders were created through explicit instructions to simulate poor coding practices. Although our metrics successfully identified these simulated cases, further

research is needed to validate their diagnostic capabilities in real-world scenarios. This includes testing the method on codes generated by novice human researchers, non-expert coders, or subtly flawed AI prompts, which may produce less extreme and harder-to-detect issues than our simulated variants.

8.3 Limitation on Missing Critical Codes

Our framework evaluates coders relative to the Aggregated Code Space (ACS), which is constructed from the combined outputs of all coders. As a result, it cannot detect critical codes that are entirely absent from every individual code space. This blind spot is not unique to our method but inherent to all team-based or consensus-oriented qualitative evaluation approaches: if no coder identifies a particular concept, it will not appear in the collective representation. Ensuring coder diversity and using multiple human or model perspectives remain essential to mitigate this limitation.

8.4 Limited Interpretations on Deviant LLM Behaviors

Despite 3 out of 4 models behaving similarly in our experiment (Hypothesis 3), our findings observed deviated merging behaviors and behaviors with Gemini-2.5-Pro, which showed a tendency to merge codes more. Even when the ranking order is relatively stable, this model choice leads to significant changes in raw metric numbers. Further research should investigate whether and/or why the stronger reasoning model merges differently, and whether this deep merging may prove preferable or not to human researchers.

Acknowledgments

We acknowledge the support and help from Dr. Matthew Berland, Dr. Lydia Cao, and Dr. James Spillane (and his students) during the research process. In addition, we acknowledge the support from the National Science Foundation under Grant Number 2303582.

We also acknowledge the usage of Generative AI tools within and regarding the study. With constant human supervision and quality assurance, such tools are used to assist in:

1. Software development (e.g., copilot autocompletion; refactoring; generating boilerplate code or data cleanup code).
2. Empirical data analysis (e.g., code for visualization or regression analysis based on explicit human instructions).
3. Paper writing (e.g., proofreading, editing, LaTeX syntax support).

References

Anne Adams, Peter Lunt, and Paul Cairns. 2008. A Qualitative Approach to HCI Research. In *Research Methods for Human-Computer Interaction*. Cambridge University Press.

Jennifer Attride-Stirling. 2001. Thematic networks: an analytic tool for qualitative research. *Qualitative Research*, 1(3):385–405.

Robert Bowman, Camille Nadal, Kellie Morrissey, Anja Thieme, and Gavin Doherty. 2023. Using Thematic Analysis in Healthcare HCI at CHI: A Scoping Review. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–18, Hamburg Germany. ACM.

Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101.

Virginia Braun and Victoria Clarke. 2012. *Thematic analysis.* American Psychological Association.

Virginia Braun and Victoria Clarke. 2013. Successful qualitative research: A practical guide for beginners.

Virginia Braun and Victoria Clarke. 2021. One size fits all? What counts as quality practice in (reflexive) thematic analysis? *Qualitative Research in Psychology*, 18(3):328–352.

Joy D. Bringer, Lynne H. Johnston, and Celia H. Brackenridge. 2004. Maximizing Transparency in a Doctoral Thesis1: The Complexities of Writing About the Use of QSR*NVIVO Within a Grounded Theory Study. *Qualitative Research*, 4(2):247–265.

M. Ariel Cascio, Eunlye Lee, Nicole Vaudrin, and Darcy A. Freedman. 2019. A Team-based Approach to Open Coding: Considerations for Creating Intercoder Consensus. *Field Methods*, 31(2):116–130.

John Chen, Alexandros Lotsos, Grace Wang, Lexie Zhao, Bruce Sherin, Uri Wilensky, and Michael Horn. 2025. Processes matter: How ml/gai approaches could support open qualitative coding of online discourse datasets. In *Proceedings of the 18th International Conference on Computer-Supported Collaborative Learning-CSCL 2025, pp. 415-419*. International Society of the Learning Sciences.

Juliet Corbin and Anselm Strauss. 2008a. Chapter 10 / Analyzing Data for Concepts. In *Basics of Qualitative Research (3rd ed.): Techniques and Procedures*

for Developing Grounded Theory. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States.

Juliet Corbin and Anselm Strauss. 2008b. Chapter 14 / Criteria for Evaluation. In *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States.

Juliet M. Corbin and Anselm Strauss. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology*, 13(1):3–21. Publisher: Springer.

Shih-Chieh Dai, Aiping Xiong, and Lun-Wei Ku. 2023. LLM-in-the-loop: Leveraging large language model for thematic analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9993–10001, Singapore. Association for Computational Linguistics.

Stefano De Paoli. 2023a. Can Large Language Models emulate an inductive Thematic Analysis of semi-structured interviews? An exploration and provocation on the limits of the approach and the model. *arXiv preprint*. ArXiv:2305.13014 [cs].

Stefano De Paoli. 2023b. Performing an Inductive Thematic Analysis of Semi-Structured Interviews With a Large Language Model: An Exploration and Provocation on the Limits of the Approach. *Social Science Computer Review*, 0(0):1–23.

Dominic Furniss, Ann Blandford, and Paul Curzon. 2011. Confessions from a grounded theory PhD: experiences and lessons learnt. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 113–122, Vancouver BC Canada. ACM.

Simret Araya Gebreegziabher, Zheng Zhang, Xiaohang Tang, Yihao Meng, Elena L. Glassman, and Toby Jia-Jun Li. 2023. PaTAT: Human-AI Collaborative Qualitative Coding with Explainable Interactive Rule Synthesis. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, pages 1–19, New York, NY, USA. Association for Computing Machinery.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Jasy Suet Yan Liew, Nancy McCracken, Shichun Zhou, and Kevin Crowston. 2014. Optimizing Features in Active Machine Learning for Complex Qualitative Content Analysis. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 44–48, Baltimore, MD, USA. Association for Computational Linguistics.

Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–23.

Angelina Parfenova, Andreas Marfurt, Jürgen Pfeffer, and Alexander Denzler. 2025. Text annotation via inductive coding: Comparing human experts to llms in qualitative data analysis. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6456–6469.

Hamed Rahimi, Jacob Louis Hoover, David Mimno, Hubert Naacke, Camelia Constantin, and Bernd Amann. 2023. Contextualized topic coherence metrics. *arXiv preprint arXiv:2305.14587*.

Md Shidur Rahman. 2016. The Advantages and Disadvantages of Using Qualitative and Quantitative Approaches and Methods in Language "Testing and Assessment" Research: A Literature Review. *Journal of Education and Learning*, 6(1):102.

Tim Rietz and Alexander Maedche. 2021. Cody: An AI-Based System to Semi-Automate Coding for Qualitative Research. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–14, New York, NY, USA. Association for Computing Machinery.

Sina Mahdipour Saravani, Sadaf Ghaffari, Yanye Luther, James Folkestad, and Marcia Moraes. 2023. Automated code extraction from discussion. In *Advances in Quantitative Ethnography: 4th International Conference, ICQE 2022, Copenhagen, Denmark, October 15–19, 2022, Proceedings*, page 227. Springer Nature.

Benjamin Saunders, Julius Sim, Tom Kingstone, Shula Baker, Jackie Waterfield, Bernadette Bartlam, Heather Burroughs, and Clare Jinks. 2018. Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & Quantity*, 52(4):1893–1907.

Carson Sievert and Kenneth Shirley. 2014. Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70.

Ravi Sinha, Idris Solola, Ha Nguyen, Hillary Swanson, and LuEttaMae Lawrence. 2024. The Role of Generative AI in Qualitative Research: GPT-4's Contributions to a Grounded Theory Analysis. In *Proceedings of the Symposium on Learning, Design and Technology*, pages 17–25, Delft Netherlands. ACM.

Cesare Spinoso-Di Piano. 2023. *Qualitative code suggestion: A human-centric approach to qualitative coding*. McGill University (Canada).

Anselm Strauss and Juliet Corbin. 1998. Basics of qualitative research: Techniques and procedures for developing grounded theory, 2nd ed. *Basics of qualitative research: Techniques and procedures for developing grounded theory, 2nd ed.*, pages xiii, 312–xiii, 312. Place: Thousand Oaks, CA, US Publisher: Sage Publications, Inc.

Gemma Team. 2025a. Gemma 3.

Qwen Team. 2025b. Qwq-32b: Embracing the power of reinforcement learning.

Gareth Terry, Nikki Hayfield, Victoria Clarke, and Virginia Braun. 2017. Thematic analysis. *The SAGE handbook of qualitative research in psychology*, 2(17-37):25. Publisher: SAGE Publications Ltd.

David R. Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation*, 27(2):237–246.

Anthony G. Tuckett. 2005. Applying thematic analysis theory to practice: A researcher's experience. *Contemporary Nurse*, 19(1-2):75–87.

Ziang Xiao, Xingdi Yuan, Q. Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. 2023. Supporting Qualitative Analysis with Large Language Models: Combining Codebook with GPT-3 for Deductive Coding. In *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23 Companion, pages 75–78, New York, NY, USA. Association for Computing Machinery.

Andres Felipe Zambrano, Xiner Liu, Amanda Barany, Ryan S. Baker, Juhan Kim, and Nidhi Nasiar. 2023. From nCoder to ChatGPT: From Automated Coding to Refining Human Coding. In *Advances in Quantitative Ethnography*, Communications in Computer and Information Science, pages 470–485, Cham. Springer Nature Switzerland.

Fengxiang Zhao, Fan Yu, and Yi Shang. 2024. A new method supporting qualitative data analysis through prompt generation for inductive coding. In *2024 IEEE International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 164–169. IEEE.

A Potential Risks

While our computational method is designed to mitigate risks in human-AI qualitative analysis, its improper application or the misinterpretation of its metrics can introduce other potential risks.

A.1 Misinterpretation of Metrics

Especially when coming from a non-qualitative research background, users may treat our computational metrics as objective, definitive measures of quality, contrary to the exploratory and subjective nature of inductive coding. For example, when substituting a baseline machine coder with its deviant variants, we interpreted the higher Divergence metrics as an indicator of potential hallucination. However, deviating from the consensus can also be valuable in other scenarios, such as a coder from a different intellectual tradition or lived background. Over-reliance on metric values can prevent junior researchers from developing the critical, interpretive skills that come from manual comparison, reflection, and building consensus with peers. Hence, our metrics should be used as diagnostic tools to prompt deeper qualitative inquiry, not as a substitute for it.

A.2 Dependency on the Coding Team

The Aggregated Code Space (ACS) is fundamentally a synthesis of the input codebook group. The quality and comprehensiveness of the evaluation are therefore dependent on the diversity and rigor of the coding team. If the entire human-AI team shares a particular bias or overlooks a critical theme, the ACS will reflect this omission. Consequently, researchers who rely solely on metric values risk automation bias, which diminishes critical engagement with the raw data and the coding process. We encourage researchers to continue recruiting a diverse human team and employing multiple LLMs for inductive coding. Moreover, they should also explore the interactive interface generated by our software package, which provides a network-based visualization of codes and metrics. We intend to further study the interface in an upcoming research project.

A.3 Privacy and Data Security

Our method involves processing data through Large Language Models and text embedding models. While our study prioritized the use of local, open-source models to protect sensitive data, researchers applying this method with proprietary, the usage of cloud-based APIs risk exposing confidential or personally identifiable information from their datasets. It is hence critical for researchers to maintain adherence to IRB protocols and relevant data privacy regulations.

B Appendices

B.1 Regression Results for Hypothesis 1

This appendix provides complete regression tables supporting **Hypothesis 1** (Section 5.1.1), which tests whether the algorithmic condition significantly affects the number of consolidated codes and the quality of generated open codes.

We report outputs from five ordinary least squares (OLS) models. **Regression 1** models the number of consolidated codes per transcript as a function of condition. **Regression 2** consists of four separate models, each predicting one code-level metric: coverage, overlap, novelty, or divergence. Conditions are dummy-coded, and coder identity is sum-coded. All regressions use the design matrix described in Section 5.1.1, with coder identity sum-coded, condition dummy-coded (Condition 1 as the baseline), and heteroscedasticity-robust (HC3) standard errors.

Table 2: Regression output for **Consolidated Code Count** (Hypothesis 1).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	1509.00	–	–	–	[–, –]
Condition 2	-54.00	–	–	–	[–, –]
Condition 3	-110.88	3.01	-36.90	0.000	[-116.77, -104.99]
Condition 4	-651.53	16.97	-38.40	0.000	[-684.78, -618.27]

Note: Standard errors unavailable for some parameters due to rank deficiency in constraints.

Table 3: Regression output for **Coverage %** (Hypothesis 1).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	28.89	0.80	34.20	0.000	[27.20, 30.50]
Condition 2	0.09	1.20	0.08	0.936	[-2.20, 2.40]
Condition 3	3.60	0.90	4.23	0.000	[1.90, 5.30]
Condition 4	7.02	0.87	8.06	0.000	[5.30, 8.70]
(Coder dummies omitted)					

Table 4: Regression output for **Overlap %** (Hypothesis 1).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	15.81	1.10	13.93	0.000	[13.60, 18.00]
Condition 2	-0.09	1.60	-0.06	0.956	[-3.30, 3.10]
Condition 3	5.45	1.10	4.75	0.000	[3.20, 7.70]
Condition 4	7.86	1.20	6.69	0.000	[5.60, 10.20]
(Coder dummies omitted)					

Table 5: Regression output for **Novelty %** (Hypothesis 1).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	23.95	0.20	97.69	0.000	[23.50, 24.40]
Condition 2	0.05	0.30	0.15	0.879	[-0.60, 0.70]
Condition 3	0.94	0.26	3.61	0.000	[0.40, 1.40]
Condition 4	-1.64	0.27	-6.10	0.000	[-2.20, -1.10]
(Coder dummies omitted)					

Table 6: Regression output for **Divergence %** (Hypothesis 1).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	72.35	0.60	125.55	0.000	[71.20, 73.50]
Condition 2	0.37	0.80	0.48	0.633	[-1.10, 1.90]
Condition 3	-4.31	0.59	-7.29	0.000	[-5.50, -3.20]
Condition 4	-1.91	0.60	-3.20	0.001	[-3.10, -0.70]
(Coder dummies omitted)					

B.2 Coder Rankings for Hypothesis 2

This appendix supports **Hypothesis 2** (Section 5.1.2), which examines the relative ranking of coders across evaluation conditions. For each metric, we present the metric outcomes and rankings of 8 coders (plus two groups) per condition. **Rel** denotes the strength of evidence that the listed coder performs better than the next one in the ranking chain.

Symbols: $>>>$ $p \leq 0.001$; $>>$ $0.001 < p \leq 0.01$; $>$ $0.01 < p \leq 0.05$; $\approx$ $p > 0.05$. These thresholds are derived from Tukey's HSD post-hoc comparisons following a one-way ANOVA on metric values, conducted separately for each condition. **Special note for Conditions 1 and 2**: those conditions do not involve probabilistic LLM-based merging and therefore are deterministic, denoted with $>>>$.

Table 7: Coder rankings by condition for **Coverage** (Hypothesis 2).

Condition 1	Rel	Condition 2	Rel	Condition 3	Rel	Condition 4	Rel
group: ai (84.80%)	>>>	group: ai (84.85%)	>>>	group: ai (87.66%)	>>>	group: ai (87.29%)	>>>
item-verb-ai (48.25%)	>>>	item-any-ai (48.46%)	>>>	item-any-ai (56.67%)	>>>	item-any-ai (60.48%)	>>>
item-any-ai (47.92%)	>>>	item-verb-ai (48.24%)	>>>	item-verb-ai (52.77%)	>>>	item-verb-ai (57.11%)	>>>
group: human (40.58%)	>>>	group: human (40.22%)	>>>	group: human (43.93%)	>>>	group: human (47.10%)	>>>
human-c (15.15%)	>>>	human-c (15.16%)	>>>	human-c (18.36%)	>>>	human-c (23.83%)	>>>
human-b (14.10%)	>>>	human-b (13.90%)	>>>	human-b (16.11%)	>>>	human-b (20.36%)	>>>
human-a (10.65%)	>>>	human-a (10.96%)	>>>	chunk-structured-ai (13.68%)	>	chunk-structured-ai (17.32%)	≈
human-d (10.18%)	>>>	human-d (10.34%)	>>>	chunk-barany-ai (12.64%)	≈	chunk-barany-ai (16.60%)	>
chunk-barany-ai (8.90%)	>>>	chunk-barany-ai (8.89%)	>>>	human-d (12.43%)	>>>	human-d (15.25%)	>
chunk-structured-ai (8.38%)		chunk-structured-ai (8.83%)		human-a (10.71%)		human-a (13.83%)	

Table 8: Coder rankings by condition for **Overlap** (Hypothesis 2).

Condition 1	Rel	Condition 2	Rel	Condition 3	Rel	Condition 4	Rel
group: ai (51.59%)	>>>	group: ai (50.79%)	>>>	group: ai (60.09%)	>>>	group: ai (57.30%)	>>>
item-any-ai (29.22%)	>>>	item-any-ai (29.28%)	>>>	item-any-ai (41.64%)	>>>	item-any-ai (44.60%)	>>>
item-verb-ai (28.09%)	>>>	item-verb-ai (27.88%)	>>>	item-verb-ai (36.49%)	>>>	item-verb-ai (40.72%)	>>>
group: human (16.96%)	>>>	group: human (16.62%)	>>>	group: human (23.33%)	>>>	group: human (24.71%)	>>>
human-b (7.19%)	>>>	human-b (6.97%)	>>>	human-c (10.65%)	≈	human-c (15.25%)	>
human-c (6.94%)	>>>	human-c (6.89%)	>>>	human-b (9.86%)	≈	human-b (13.14%)	≈
human-a (5.91%)	>>>	human-a (6.06%)	>>>	chunk-structured-ai (8.60%)	≈	chunk-structured-ai (11.12%)	≈
human-d (5.34%)	>>>	human-d (5.42%)	>>>	human-d (7.73%)	≈	chunk-barany-ai (10.83%)	≈
chunk-structured-ai (3.45%)	>>>	chunk-structured-ai (3.73%)	>>>	chunk-barany-ai (7.68%)	≈	human-d (10.03%)	≈
chunk-barany-ai (3.40%)		chunk-barany-ai (3.56%)		human-a (6.57%)		human-a (8.97%)	

Table 9: Coder rankings by condition for **Novelty** (Hypothesis 2).

Condition 1	Rel	Condition 2	Rel	Condition 3	Rel	Condition 4	Rel
group: ai (80.50%)	>>>	group: ai (80.44%)	>>>	group: ai (81.03%)	>>>	group: ai (77.74%)	>>>
item-any-ai (42.43%)	>>>	item-any-ai (42.20%)	>>>	item-any-ai (45.05%)	>>>	item-any-ai (36.59%)	≈
item-verb-ai (39.34%)	>>>	item-verb-ai (39.69%)	>>>	item-verb-ai (42.12%)	>>>	item-verb-ai (35.80%)	>>>
group: human (30.17%)	>>>	group: human (30.25%)	>>>	group: human (30.84%)	>>>	group: human (27.12%)	>>>
human-c (11.65%)	>>>	human-c (11.49%)	>>>	human-c (12.84%)	>>>	human-c (11.78%)	>>>
human-b (9.43%)	>>>	human-b (9.46%)	>>>	human-b (9.77%)	>>>	human-b (8.47%)	≈
chunk-structured-ai (7.44%)	>>>	chunk-structured-ai (7.54%)	>>>	chunk-structured-ai (7.79%)	>	chunk-structured-ai (8.00%)	≈
chunk-barany-ai (7.20%)	>>>	chunk-barany-ai (7.21%)	>>>	chunk-barany-ai (7.21%)	>>>	chunk-barany-ai (7.36%)	>>>
human-d (6.14%)	>>>	human-d (6.25%)	>>>	human-d (6.44%)	>	human-d (5.37%)	≈
human-a (5.25%)		human-a (5.54%)		human-a (5.82%)		human-a (4.92%)	

Table 10: Coder rankings by condition for **Divergence** (Hypothesis 2).

Condition 1	Rel	Condition 2	Rel	Condition 3	Rel	Condition 4	Rel
chunk-barany-ai (78.87%)	>>>	chunk-barany-ai (78.70%)	>>>	human-a (73.88%)	>	human-a (74.57%)	≈
chunk-structured-ai (78.50%)	>>>	chunk-structured-ai (78.38%)	>>>	chunk-barany-ai (72.87%)	≈	human-d (73.98%)	≈
human-d (75.40%)	>>>	human-d (75.69%)	>>>	human-d (72.61%)	>	chunk-barany-ai (73.59%)	≈
human-c (74.87%)	>>>	human-c (75.31%)	>>>	chunk-structured-ai (71.57%)	≈	chunk-structured-ai (73.41%)	>>
human-a (74.72%)	>>>	human-a (75.10%)	>>>	human-c (71.24%)	≈	human-b (72.37%)	≈
human-b (74.09%)	>>>	human-b (74.67%)	>>>	human-b (70.89%)	>>>	human-c (71.86%)	>>>
group: human (69.45%)	>>>	group: human (69.94%)	>>>	group: ai (65.24%)	≈	group: ai (70.05%)	>>>
group: ai (68.84%)	>>>	group: ai (69.45%)	>>>	group: human (64.77%)	>>>	group: human (68.09%)	>>>
item-verb-ai (65.08%)	>>>	item-verb-ai (65.56%)	>>>	item-verb-ai (60.17%)	>>>	item-verb-ai (64.07%)	>>>
item-any-ai (63.72%)		item-any-ai (64.43%)		item-any-ai (57.20%)		item-any-ai (62.41%)	

B.3 Regression Results for Hypotheses 3 and 4

This appendix providesOLS regression results supporting **Hypotheses 3** and **4** (Section 5.2.1 and 5.2.2). Hypothesis 3 tests whether the LLM used in the merging process significantly influences each code-level metric and the number of consolidated codes. Hypothesis 4 evaluates how well condition, merging model, and coder identity together explain variation in outcomes. All results are based on the model:

$$Y \sim \text{C(condition)} + \text{C(model, Sum)} + \text{C(coder, Sum)}$$

where Y represents each outcome of interest. We use dummy-coding for the condition (Condition 3 as the baseline), sum-coding for the model and coder identity (Average as the baseline), and HC3 heteroscedasticity-robust standard errors. Coefficients are interpreted as deviations from the baseline.

Table 11: Regression output for **Consolidated Codes** (Hypotheses 3 and 4).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	295.2700	3.015	97.94	0.000	[289.361, 301.179]
Condition 4	-78.1800	4.442	-17.60	0.000	[-86.886, -69.474]
Gemini-2.5-pro	-15.0150	4.609	-3.26	0.001	[-24.048, -5.982]
Gemma3-27b	1.3700	3.642	0.38	0.707	[-5.768, 8.508]
GPT-4.1	2.5150	3.607	0.70	0.486	[-4.555, 9.585]
(Coder terms omitted)					

$R^2 = 0.951$ Adjusted $R^2 = 0.950$

Table 12: Regression output for **Coverage %** (Hypotheses 3 and 4).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.3250	0.001	326.50	0.000	[0.323, 0.327]
Condition 4	0.0342	0.002	22.59	0.000	[0.031, 0.037]
Gemini-2.5-pro	0.0440	0.002	24.48	0.000	[0.041, 0.048]
Gemma3-27b	-0.0122	0.001	-11.28	0.000	[-0.014, -0.010]
GPT-4.1	-0.0152	0.001	-14.75	0.000	[-0.017, -0.013]
(Coder terms omitted)					

$R^2 = 0.993$ Adjusted $R^2 = 0.993$

Table 13: Regression output for **Overlap %** (Hypotheses 3 and 4).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.2126	0.001	147.81	0.000	[0.210, 0.215]
Condition 4	0.0240	0.002	10.72	0.000	[0.020, 0.028]
Gemini-2.5-pro	0.0637	0.003	23.97	0.000	[0.058, 0.069]
Gemma3-27b	-0.0189	0.002	-11.55	0.000	[-0.022, -0.016]
GPT-4.1	-0.0213	0.001	-14.74	0.000	[-0.024, -0.018]
(Coder terms omitted)					

$R^2 = 0.970$ Adjusted $R^2 = 0.969$

Table 14: Regression output for **Novelty %** (Hypotheses 3 and 4).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.2489	0.001	309.04	0.000	[0.247, 0.251]
Condition 4	-0.0258	0.001	-20.21	0.000	[-0.028, -0.023]
Gemini-2.5-pro	0.0136	0.001	10.41	0.000	[0.011, 0.016]
Gemma3-27b	-0.0052	0.001	-4.75	0.000	[-0.007, -0.003]
GPT-4.1	-0.0046	0.001	-4.59	0.000	[-0.007, -0.003]
(Coder terms omitted)					

$R^2 = 0.994$ Adjusted $R^2 = 0.994$

Table 15: Regression output for **Divergence %** (Hypotheses 3 and 4).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.6804	0.001	803.88	0.000	[0.679, 0.682]
Condition 4	0.0240	0.001	20.54	0.000	[0.022, 0.026]
Gemini-2.5-pro	-0.0420	0.001	-31.99	0.000	[-0.045, -0.039]
Gemma3-27b	0.0121	0.001	13.04	0.000	[0.010, 0.014]
GPT-4.1	0.0166	0.001	19.77	0.000	[0.015, 0.018]
(Coder terms omitted)					

$R^2 = 0.920$ Adjusted $R^2 = 0.919$

B.4 Regression Results for Hypothesis 5

This appendix provides regression outputs supporting **Hypothesis 5** (Section 5.2.3), which evaluates the stability of metric outputs by analyzing the coefficient of variation (CoV) across stochastic LLM runs in Conditions 3 and 4. All models use ordinary least squares (OLS) regression to predict the CoV for each metric, using condition, merging model, and coder identity as predictors. Condition is dummy-coded (Condition 3 as baseline), and both model and coder identity are sum-coded (average as baseline). Robust (HC3) standard errors are used.

Table 16: Regression output for **CoV of Consolidated Code Count** (Hypothesis 5).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.0040	0.001	5.93	0.000	[0.003, 0.005]
Condition 4	0.0236	0.002	15.23	0.000	[0.021, 0.027]
Gemini-2.5-pro	0.0047	0.001	3.88	0.000	[0.002, 0.007]
Gemma3-27b	-0.0023	0.002	-1.49	0.136	[-0.005, 0.001]
GPT-4.1	0.0021	0.001	1.55	0.120	[-0.001, 0.005]
(Coder dummies omitted)					

$R^2 = 0.830$ Adjusted $R^2 = 0.797$

Table 17: Regression output for **CoV of Coverage** (Hypothesis 5).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.0272	0.003	9.36	0.000	[0.021, 0.033]
Condition 4	0.0307	0.005	6.77	0.000	[0.022, 0.040]
Gemini-2.5-pro	0.0019	0.004	0.54	0.587	[-0.005, 0.009]
Gemma3-27b	-0.0071	0.004	-1.68	0.093	[-0.015, 0.001]
GPT-4.1	0.0020	0.004	0.50	0.621	[-0.006, 0.010]
(Coder dummies omitted)					

$R^2 = 0.737$ Adjusted $R^2 = 0.686$

Table 18: Regression output for **CoV of Overlap** (Hypothesis 5).

Predictor	**coef**	**std err**	**z**	**P> |z|**	**[0.025, 0.975]**
Intercept	0.0440	0.004	11.34	0.000	[0.036, 0.052]
Condition 4	0.0490	0.007	7.45	0.000	[0.036, 0.062]
Gemini-2.5-pro	-0.0043	0.005	-0.83	0.404	[-0.014, 0.006]
Gemma3-27b	-0.0108	0.006	-1.74	0.082	[-0.023, 0.001]
GPT-4.1	0.0051	0.006	0.88	0.377	[-0.006, 0.016]
(Coder dummies omitted)					

$R^2 = 0.698$ Adjusted $R^2 = 0.639$

Table 19: Regression output for **CoV of Novelty** (Hypothesis 5).

Predictor	coef	std err	z	P> |z|	[0.025, 0.975]
Intercept	0.0359	0.004	8.09	0.000	[0.027, 0.045]
Condition 4	0.0495	0.006	7.67	0.000	[0.037, 0.062]
Gemini-2.5-pro	0.0163	0.006	2.61	0.009	[0.004, 0.029]
Gemma3-27b	-0.0130	0.006	-2.18	0.029	[-0.025, -0.001]
GPT-4.1	-0.0049	0.005	-1.01	0.311	[-0.014, 0.005]
(Coder dummies omitted)					
$R^2 = 0.784$	Adjusted $R^2 = 0.741$				

Table 20: Regression output for **CoV of Divergence** (Hypothesis 5).

Predictor	coef	std err	z	P> |z|	[0.025, 0.975]
Intercept	0.0090	0.001	17.95	0.000	[0.008, 0.010]
Condition 4	0.0038	0.001	4.80	0.000	[0.002, 0.005]
Gemini-2.5-pro	0.0042	0.001	5.61	0.000	[0.003, 0.006]
Gemma3-27b	-0.0034	0.001	-4.67	0.000	[-0.005, -0.002]
GPT-4.1	-0.0007	0.001	-1.07	0.284	[-0.002, 0.001]
(Coder dummies omitted)					
$R^2 = 0.597$	Adjusted $R^2 = 0.518$				

B.5 Coder Rankings for Hypothesis 6

This appendix presents full coder ranking tables supporting **Hypothesis 6** (Section 5.2.4), which investigates whether the choice of evaluation LLM affects the relative ranking of coders. For each metric, we compute average scores per coder and rank them separately under four evaluation models in **Condition 3** and **Condition 4**. **Rel** denotes the strength of evidence that the listed coder performs better than the next one in the ranking chain.

Symbols: $>>>$ $p \leq 0.001$; $>>$ $0.001 < p \leq 0.01$; $>$ $0.01 < p \leq 0.05$; $\approx$ $p > 0.05$. These thresholds are derived from Tukey's HSD post-hoc comparisons following a one-way ANOVA on metric values, conducted separately for each condition.

Table 21: Coder rankings for **Coverage**, Condition 3 (Hypothesis 6).

Condition 3	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
group: ai (87.66%)	>>>	group: ai (90.23%)	>>>	group: ai (86.50%)	>>>	group: ai (86.64%)	>>>	group: ai (87.25%)	>>>
item-any-ai (56.67%)	>>>	item-any-ai (60.27%)	>	item-any-ai (56.54%)	>>>	item-any-ai (54.61%)	>>>	item-any-ai (55.27%)	>>>
item-verb-ai (52.77%)	>>>	item-verb-ai (58.96%)	>>>	item-verb-ai (51.35%)	>>>	item-verb-ai (49.89%)	>>>	item-verb-ai (50.87%)	>>>
group: human (43.93%)	>>>	group: human (48.08%)	>>>	group: human (42.39%)	>>>	group: human (42.34%)	>>>	group: human (42.90%)	>>>
human-c (18.36%)	>>>	human-c (22.85%)	>>>	human-c (16.47%)	≈	human-c (16.79%)	>>>	human-c (17.32%)	>>>
human-b (16.11%)	>>>	human-b (18.11%)	>>>	human-b (16.14%)	>>>	human-b (14.80%)	>>>	human-b (15.37%)	≈
chunk-structured-ai (13.68%)	>	human-d (13.64%)	≈	chunk-structured-ai (14.29%)	>>>	chunk-structured-ai (12.71%)	≈	chunk-structured-ai (14.69%)	>>>
chunk-barany-ai (12.64%)	≈	chunk-structured-ai (13.02%)	≈	chunk-barany-ai (12.83%)	>>>	chunk-barany-ai (11.83%)	≈	chunk-barany-ai (13.15%)	≈
human-d (12.43%)	>>>	chunk-barany-ai (12.74%)	>	human-d (12.02%)	>>>	human-d (11.62%)	>>	human-d (12.44%)	>>>
human-a (10.71%)		human-a (11.39%)		human-a (10.40%)		human-a (10.52%)		human-a (10.55%)	

Table 22: Coder rankings for **Coverage**, Condition 4 (Hypothesis 6).

Condition 4	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
group: ai (87.29%)	>>>	group: ai (91.26%)	>>>	group: ai (85.28%)	>>>	group: ai (87.15%)	>>>	group: ai (85.45%)	>>>
item-any-ai (60.48%)	>>>	item-any-ai (69.10%)	>>>	item-any-ai (56.87%)	>>>	item-any-ai (59.19%)	>>>	item-any-ai (56.76%)	>>>
item-verb-ai (57.11%)	>>>	item-verb-ai (65.58%)	>>>	item-verb-ai (54.42%)	>>>	item-verb-ai (54.93%)	>>>	item-verb-ai (53.49%)	>>>
group: human (47.10%)	>>>	group: human (55.52%)	>>>	group: human (44.90%)	>>>	group: human (44.16%)	>>>	group: human (43.83%)	>>>
human-c (23.83%)	>>>	human-c (32.22%)	>>>	human-c (20.99%)	>>	human-c (21.81%)	>>>	human-c (20.30%)	>>>
human-b (20.36%)	>>>	human-b (27.30%)	>>>	human-b (18.91%)	>	human-b (18.06%)	>>>	human-b (17.17%)	≈
chunk-structured-ai (17.32%)	≈	chunk-structured-ai (21.35%)	≈	chunk-structured-ai (16.97%)	≈	chunk-structured-ai (15.32%)	≈	chunk-structured-ai (15.66%)	≈
chunk-barany-ai (16.60%)	>	chunk-barany-ai (21.32%)	≈	chunk-barany-ai (15.87%)	>	chunk-barany-ai (14.95%)	≈	chunk-barany-ai (14.27%)	≈
human-d (15.25%)	>	human-d (20.43%)	≈	human-d (13.96%)	≈	human-d (14.04%)	≈	human-d (12.56%)	≈
human-a (13.83%)		human-a (18.83%)		human-a (12.67%)		human-a (12.43%)		human-a (11.38%)	

Table 23: Coder rankings for **Overlap**, Condition 3 (Hypothesis 6).

Condition 3	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
group: ai (60.09%)	>>>	group: ai (70.62%)	>>>	group: ai (55.60%)	>>>	group: ai (56.08%)	>>>	group: ai (58.08%)	>>>
item-any-ai (41.64%)	>>>	item-any-ai (47.70%)	>>	item-any-ai (41.21%)	>>>	item-any-ai (38.03%)	>>>	item-any-ai (39.60%)	>>>
item-verb-ai (36.49%)	>>>	item-verb-ai (45.72%)	>>>	item-verb-ai (34.58%)	>>>	item-verb-ai (31.84%)	>>>	item-verb-ai (33.82%)	>>>
group: human (23.33%)	>>>	group: human (28.80%)	>>>	group: human (21.55%)	>>>	group: human (20.88%)	>>>	group: human (22.08%)	>>>
human-c (10.65%)	≈	human-c (15.43%)	>>>	human-b (9.76%)	≈	human-c (8.78%)	≈	human-c (9.57%)	≈
human-b (9.86%)	≈	human-b (12.21%)	>>>	chunk-structured-ai (9.14%)	≈	human-b (8.28%)	≈	chunk-structured-ai (9.48%)	≈
chunk-structured-ai (8.60%)	≈	human-d (9.20%)	≈	human-c (8.82%)	>>	chunk-structured-ai (7.44%)	≈	human-b (9.18%)	≈
human-d (7.73%)	≈	chunk-structured-ai (8.37%)	≈	chunk-barany-ai (7.85%)	≈	human-d (6.77%)	≈	chunk-barany-ai (8.06%)	≈
chunk-barany-ai (7.68%)	≈	chunk-barany-ai (8.14%)	≈	human-d (7.34%)	>>>	chunk-barany-ai (6.68%)	≈	human-d (7.62%)	>
human-a (6.57%)		human-a (7.55%)		human-a (6.19%)		human-a (6.22%)		human-a (6.30%)	

Table 24: Coder rankings for **Overlap**, Condition 4 (Hypothesis 6).

Condition 4	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
group: ai (57.30%)	>>>	group: ai (73.02%)	>>>	group: ai (50.65%)	>>>	group: ai (55.45%)	>>>	group: ai (50.09%)	>>>
item-any-ai (44.60%)	>>>	item-any-ai (57.66%)	>>	item-any-ai (39.58%)	>>>	item-any-ai (42.19%)	>>>	item-any-ai (38.97%)	>>>
item-verb-ai (40.72%)	>>>	item-verb-ai (54.01%)	>>>	item-verb-ai (36.58%)	>>>	item-verb-ai (37.39%)	>>>	item-verb-ai (34.92%)	>>>
group: human (24.71%)	>>>	group: human (34.62%)	>>>	group: human (21.65%)	>>>	group: human (21.66%)	>>>	group: human (20.90%)	>>>
human-c (15.25%)	>	human-c (24.31%)	>>>	human-c (12.09%)	≈	human-c (13.18%)	≈	human-c (11.40%)	≈
human-b (13.14%)	≈	human-b (20.23%)	>>>	human-b (11.52%)	≈	human-b (10.89%)	≈	human-b (9.92%)	≈
chunk-structured-ai (11.12%)	≈	chunk-barany-ai (15.39%)	≈	chunk-structured-ai (10.89%)	≈	chunk-barany-ai (9.10%)	≈	chunk-structured-ai (9.72%)	≈
chunk-barany-ai (10.83%)	≈	human-d (15.11%)	≈	chunk-barany-ai (10.15%)	≈	chunk-structured-ai (9.01%)	≈	chunk-barany-ai (8.68%)	≈
human-d (10.03%)	≈	chunk-structured-ai (14.88%)	≈	human-d (8.65%)	≈	human-d (8.98%)	≈	human-d (7.37%)	≈
human-a (8.97%)		human-a (13.66%)		human-a (7.78%)		human-a (7.81%)		human-a (6.62%)	

Table 25: Coder rankings for **Novelty**, Condition 3 (Hypothesis 6).

Condition 3	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
group: ai (81.03%)	>>>	group: ai (83.66%)	>>>	group: ai (79.24%)	>>>	group: ai (80.56%)	>>>	group: ai (80.67%)	>>>
item-any-ai (45.05%)	>>>	item-any-ai (47.25%)	>>>	item-any-ai (44.34%)	>>>	item-any-ai (44.33%)	>>>	item-any-ai (44.26%)	>>>
item-verb-ai (42.12%)	>>>	item-verb-ai (44.94%)	>>>	item-verb-ai (41.07%)	>>>	item-verb-ai (40.69%)	>>>	item-verb-ai (41.78%)	>>>
group: human (30.84%)	>>>	group: human (31.36%)	>>>	group: human (30.60%)	>>>	group: human (30.79%)	>>>	group: human (30.61%)	>>>
human-c (12.84%)	>>>	human-c (13.54%)	>>>	human-c (12.44%)	>>>	human-c (12.60%)	>>>	human-c (12.79%)	>>>
human-b (9.77%)	>>>	human-b (9.93%)	>>	human-b (9.76%)	>>>	human-b (9.97%)	>>>	human-b (9.44%)	>>
chunk-structured-ai (7.79%)	>	chunk-structured-ai (8.68%)	≈	chunk-structured-ai (7.50%)	>>	chunk-structured-ai (7.65%)	≈	chunk-structured-ai (7.34%)	≈
chunk-barany-ai (7.21%)	>>>	chunk-barany-ai (7.90%)	>	chunk-barany-ai (6.86%)	≈	chunk-barany-ai (7.08%)	>>>	chunk-barany-ai (6.99%)	≈
human-d (6.44%)	>	human-d (6.77%)	≈	human-d (6.44%)	>	human-d (6.09%)	≈	human-d (6.46%)	≈
human-a (5.82%)		human-a (5.79%)		human-a (5.94%)		human-a (5.76%)		human-a (5.80%)	

Table 26: Coder rankings for **Novelty**, Condition 4 (Hypothesis 6).

Condition 4	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
group: ai (77.74%)	>>>	group: ai (80.88%)	>>>	group: ai (75.22%)	>>>	group: ai (78.34%)	>>>	group: ai (76.50%)	>>>
item-any-ai (36.59%)	≈	item-any-ai (39.56%)	>>>	item-verb-ai (34.99%)	≈	item-any-ai (36.49%)	>	item-verb-ai (36.74%)	≈
item-verb-ai (35.80%)	>>>	item-verb-ai (36.31%)	>>>	item-any-ai (34.74%)	>>>	item-verb-ai (35.17%)	>>>	item-any-ai (35.58%)	>>>
group: human (27.12%)	>>>	group: human (27.41%)	>>>	group: human (28.28%)	>>>	group: human (25.06%)	>>>	group: human (27.73%)	>>>
human-c (11.78%)	>>>	human-c (11.97%)	≈	human-c (11.84%)	>>>	human-c (11.14%)	>>>	human-c (12.16%)	>>>
human-b (8.47%)	≈	chunk-structured-ai (11.29%)	≈	human-b (8.70%)	>>>	chunk-structured-ai (8.26%)	≈	human-b (8.03%)	>>
chunk-structured-ai (8.00%)	≈	chunk-barany-ai (9.62%)	≈	chunk-barany-ai (6.97%)	≈	human-b (7.58%)	≈	chunk-structured-ai (6.15%)	≈
chunk-barany-ai (7.36%)	>>>	human-b (9.59%)	>>>	chunk-structured-ai (6.31%)	≈	chunk-barany-ai (7.15%)	>>>	chunk-barany-ai (5.69%)	≈
human-d (5.37%)	≈	human-d (6.59%)	≈	human-d (5.41%)	≈	human-d (4.16%)	≈	human-d (5.33%)	≈
human-a (4.92%)		human-a (6.24%)		human-a (5.04%)		human-a (4.09%)		human-a (4.33%)	

Table 27: Coder rankings for **Divergence**, Condition 3 (Hypothesis 6).

Condition 3	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
human-a (73.88%)	>	human-a (71.91%)	≈	human-a (74.56%)	>>>	human-a (74.62%)	≈	human-a (74.44%)	>>>
chunk-barany-ai (72.87%)	≈	chunk-barany-ai (71.78%)	≈	human-c (73.38%)	≈	chunk-barany-ai (74.52%)	≈	human-d (72.83%)	≈
human-d (72.61%)	>	chunk-structured-ai (71.29%)	≈	human-d (73.05%)	≈	human-d (74.13%)	≈	chunk-barany-ai (72.40%)	≈
chunk-structured-ai (71.57%)	≈	human-d (70.45%)	>>>	chunk-barany-ai (72.76%)	>>>	chunk-structured-ai (73.53%)	≈	human-c (72.38%)	≈
human-c (71.24%)	≈	human-b (67.73%)	>>>	human-b (71.07%)	≈	human-c (73.43%)	≈	human-b (71.78%)	>>>
human-b (70.89%)	>>>	human-c (65.79%)	>>>	chunk-structured-ai (71.01%)	>>>	human-b (72.96%)	>>>	chunk-structured-ai (70.44%)	>>>
group: ai (65.24%)	≈	group: ai (60.48%)	≈	group: ai (66.89%)	>	group: ai (67.14%)	≈	group: ai (66.46%)	≈
group: human (64.77%)	>>>	group: human (59.93%)	>>>	group: human (66.34%)	>>>	group: human (66.88%)	>>>	group: human (65.93%)	>>>
item-verb-ai (60.17%)	>>>	item-verb-ai (53.78%)	>>	item-verb-ai (61.38%)	>>>	item-verb-ai (63.30%)	>>>	item-verb-ai (62.21%)	>>>
item-any-ai (57.20%)		item-any-ai (52.27%)		item-any-ai (57.83%)		item-any-ai (59.91%)		item-any-ai (58.81%)	

Table 28: Coder rankings for **Divergence**, Condition 4 (Hypothesis 6).

Condition 4	Rel	Gemini-2.5-pro	Rel	Gemma3-27B	Rel	GPT-4.1	Rel	Qwen-QwQ-32B	Rel
human-a (74.57%)	≈	chunk-structured-ai (70.94%)	≈	human-a (75.75%)	≈	human-a (75.38%)	≈	human-a (76.45%)	≈
human-d (73.98%)	≈	human-a (70.68%)	≈	human-d (75.28%)	≈	chunk-structured-ai (75.00%)	≈	human-d (75.98%)	≈
chunk-barany-ai (73.59%)	≈	chunk-barany-ai (70.36%)	≈	human-c (74.41%)	≈	chunk-barany-ai (74.76%)	≈	chunk-barany-ai (75.01%)	≈
chunk-structured-ai (73.41%)	>>	human-d (70.09%)	>>>	chunk-barany-ai (74.23%)	≈	human-d (74.59%)	≈	human-b (74.54%)	≈
human-b (72.37%)	≈	human-b (67.20%)	>	human-b (73.84%)	≈	human-b (73.89%)	≈	human-c (74.33%)	≈
human-c (71.86%)	>>>	human-c (65.45%)	≈	chunk-structured-ai (73.53%)	>>	human-c (73.23%)	>>>	chunk-structured-ai (74.18%)	>>>
group: ai (70.05%)	>>>	group: ai (65.00%)	>>>	group: ai (72.02%)	>>>	group: ai (71.33%)	>>	group: ai (71.83%)	>
group: human (68.09%)	>>>	group: human (61.91%)	>>>	group: human (70.45%)	>>>	group: human (69.62%)	>>>	group: human (70.40%)	>>>
item-verb-ai (64.07%)	>>>	item-verb-ai (57.50%)	≈	item-verb-ai (66.28%)	>>	item-verb-ai (65.87%)	>>>	item-verb-ai (66.62%)	>>>
item-any-ai (62.41%)		item-any-ai (56.40%)		item-any-ai (64.94%)		item-any-ai (63.87%)		item-any-ai (64.45%)	