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

JOHN CHEN, Northwestern University, United States of America

ALEXANDROS LOTSOS, Northwestern University, United States of America

CAIYI WANG, Northwestern University, United States of America

LEXIE ZHAO, Northwestern University, United States of America

JESSICA HULLMAN, Northwestern University, United States of America

BRUCE L. SHERIN, Northwestern University, United States of America

URI J. WILENSKY, Northwestern University, United States of America

MICHAEL S. HORN, Northwestern University, United States of America

Qualitative analysis is critical to understanding human datasets in many social science disciplines. Open coding is an inductive qualitative process that identifies and interprets "open codes" from datasets. Yet, meeting methodological expectations (such as "as exhaustive as possible") can be challenging. While many machine learning (ML)/generative AI (GAI) studies have attempted to support open coding, few have systematically measured or evaluated GAI outcomes, increasing potential bias risks. Building on Grounded Theory and Thematic Analysis theories, we present a computational method to measure and identify potential biases from "open codes" systematically. Instead of operationalizing human expert results as the "ground truth," our method is built upon a team-based approach between human and machine coders. We experiment with two HCI datasets to establish this method's reliability by 1) comparing it with human analysis, and 2) analyzing its output stability. We present evidence-based suggestions and example workflows for ML/GAI to support open coding.

1 INTRODUCTION

Qualitative analysis is the process of systematically identifying, generating, and organizing concepts from data. It has been widely adopted in many social science disciplines (such as education, sociology, psychology, and medicine), as well as interdisciplinary areas such as human-computer interaction (HCI) [1] to understand people's perceptions, feelings, and nuanced interactions with technology[2].

However, qualitative methods can be challenging, time-consuming, and may lack transparency[9]. The problem is particularly acute during the process of **open coding**, where researchers inductively identify emergent codes from raw data without a preconceived coding scheme. As the first step of qualitative analysis, practitioners and theorists of thematic analysis (TA)[10, 63] and grounded theory (GT)[23] make frequent use of open coding and agree on its goal:

---

[1] For example, the CHI conference itself has increasingly included papers using qualitative methods: 36.7% in 2021, 48.4% in 2022, 48.9% in 2023, 53.9% in 2024. This result comes from searches in the ACM digital library with the keywords: "thematic analysis," "qualitative analysis," "qualitative coding," "qualitative research," "discourse analysis," and "grounded theory."

to capture as many aspects, patterns, or "coding moments" as possible. However, the criteria of being "open" and "as exhaustive as possible" are often inconsistent and subjective in practice[30, 58].

Machine learning (ML) and generative AI (GAI) techniques have the potential to support and improve qualitative research, but this potential has yet to be fully realized. Past studies have mostly focused on deductive coding[40, 55, 71], where researchers systematically apply existing coding schemes to new data sets. However, attempts to apply ML/GAI to open coding have been mostly exploratory. While the HCI literature repeatedly found that humans tend to over-rely on AI systems and can be easily misled when AI systems go astray[15], commercial providers (such as Atlas.ti) have already started to provide GAI solutions for open coding. Since all future analyses rely on open coding results, there is a pressing need for systematic measurement methods on open coding results (or "open codes").

We propose a theory-informed computational method to systematically measure open codes. Theorists of thematic analysis and grounded theory advocate a team-based approach[18, 24] that depends on multiple coders to find "as many relevant concepts or interpretations as possible" from the data[22, 24, 63]. Building on this concept, we computationally transformed each coder's inductive codes into a **Code Space**, and calculated the sum of all code spaces as an **Aggregated Code Space**. This enables measuring individual coders' results against a team of coders, whose combined codes may get closer to the theoretical expectation. We propose and operationalize four conceptual metrics for assessing Code Spaces: **Coverage**, **Density**, **Novelty**, and **Divergence**.

We demonstrate our method's potential as a novel computational lens to understand, compare, and evaluate open codes (from either human or machine coders). Working on two HCI datasets, we computationally measured open codes generated by five previously published ML/GAI approaches and six mainstream GAI models. We validate the reliability of our method by 1) comparing its results with human analysis, and 2) statistically analyzing its output stability. By combining machine measurement and human interpretation, we present an example of a human-AI collaborative workflow to evaluate open codes, resulting in evidence-based suggestions for qualitative researchers to adopt GAI in inductive analysis.

2 BACKGROUND

2.1 Open Qualitative Coding

Qualitative analysis enables an in-depth exploration of human experiences by focusing on the nuanced interpretations, emotions, and subjective experiences that shape individual and social realities [52]. Thematic Analysis (TA) and Grounded Theory (GT), two commonly used qualitative methods in HCI research[1, 2, 9, 21, 48, 50, 53, 54, 56], both advocate for **open coding** as an essential first step to uncover the underlying reasons and processes that drive the formation and transformation of meaning[52, 62]. However, conducting open coding can be challenging, particularly when scaling up. Ambiguity[4, 9, 66], inconsistent terminology[12], and insufficient reporting[14] can all obscure how qualitative analysis was performed, making it harder to assess its quality[9, 49]. Few studies have attempted to measure or evaluate open coding results based on theoretical expectations, further compounding the challenge.

2.1.1 Expectations of Open Coding. **Open coding** is a key approach in TA and GT that **inductively** derives codes from raw data. It can generate new coding schemes without or in addition to a theoretical framework[24, 62]. Researchers can then apply the coding scheme systematically on datasets through the **deductive coding** process[7]. With the capability of finding novel insights beyond existing knowledge, GT theorists require open coding[22, 24, 62], and TA theorists advocate for it[63]. Since subsequent analysis depends on the open coding results, without first establishing

the schemes' relative completeness, researchers risk entering a biased or unbalanced concept space that would harm the rigor of qualitative research. It is, therefore, imperative to review the expectations and challenges of open coding.

Many theorists of TA and GT agree that open coding should capture as many aspects, patterns, or "codable moments" as possible[22, 24, 63]. While open coding is not always required for TA, TA theorists such as Braun and Clarke[10, 13] argue that the focus of qualitative assurance should shift from accuracy (i.e., the consistency in deductive coding) to reflexivity (i.e., how the open coding process reflects the underlying dataset). Similarly, coming from an implicit constructivist epistemological stance, Terry et al. contend that open coding outcomes may be strong or weak, but cannot be objectively right or wrong. This shift toward reflexivity aligns with GT's inductive process, which emphasizes openness to new insights. GT stresses that researchers must continually look for emerging ideas until theoretical saturation is reached, at which point no further insights emerge[24, 47]. Since the goal is to discover potential ideas, open codes do not need to be consistent or systematic, and a single example may suffice.

2.1.2 Evaluation of Open Coding. Deriving patterns directly from raw data without a predefined framework is inherently challenging[43]. This challenge is compounded by the underdevelopment of measurement for open coding and theorists' focus on researcher quality (qualifications, experience, and perspective) over research quality[23]. Some metrics, such as validity or credibility, are easier to judge by the data or keep track of[28]. Other metrics are more elusive: for example, how can we operationalize the expectations of inductive coding - being "open" and "as exhaustive as possible"? Even when coders identify some valid and credible codes, as Corbin and Strauss acknowledge, they are only some of the "many plausible interpretations possible from data."

Some scholars have attempted to adopt deductive coding metrics such as inter-rater reliability (IRR) in open coding[44]. Yet, IRR can only assess consistency between coders. As qualitative researchers may not have or reach the **ground truth**, even complete consistency cannot measure the "openness" or "exhaustiveness" of the codes. Most qualitative researchers who adopt non-positivist epistemological stances believe that[17]: post-positivism believes in objective truth, yet humans may only approximate it[29, 37]; interpretivism argues that truth is subjective and situated[64], while constructivism holds that truth is a human construction to understand reality[37, 46]. On the other hand, while a few scholars have advocated for positivist qualitative research and believe in ground truth [6, 70], it is unclear how such truth can be reliably identified and validated.

GT theorists have explored another potential pathway of evaluation. Corbin and Strauss proposed two criteria related to inductive coding: 1) depth, which refers to the richness of descriptive detail that adds substance to findings, and 2) variation, which demonstrates the complexity of human life by incorporating diverse cases outside of dominant patterns. However, operationalizing these criteria can be difficult. Traditionally, GT theorists suggest working towards *theoretical saturation*, where no new interpretations emerge from additional data, and existing ones are all well-defined with sufficient variation[23]. Yet, in real-world research contexts, the logic for determining saturation is often inconsistent and subjective[2, 30, 58], as it is challenging to judge whether coding has truly become exhaustive [32], or has simply reached a convenient stopping point.

While the absolute saturation may be impractical, TA[18] and GT theorists[24] advocate for a team-based approach. Analyzing data from multiple perspectives reduces the likelihood of missing key codes or over-interpreting data. By constantly comparing and contrasting codes from different individuals[18], researchers can open up the analysis to the scrutiny of other researchers[24], resulting in "new insights [and] increased sensitivity" while "guarding against bias." Similarly, the GT-inspired general inductive approach[65] suggests merging independent coders' open coding

results and checking the proportion of overlapping codes. A low degree of overlap may necessitate more discussion and further analysis.

Recently, some TA scholars have attempted to quantify "saturation" by counting the number of codes that emerged throughout the coding process until the curve flattens[38, 42]. However, the work is built on (possibly overly) strong assumptions about the inductive coding process: 1) duplicated codes do not exist, and 2) researchers will consistently "sample" from all "discoverable" codes. While this line of work has practical values for predicting the number of interviews or surveys needed[42], both assumptions are unlikely to hold for human coders during inductive coding. As ML/GAI models may be systematically biased (e.g., [59, 69]), the issue is only exacerbated for machine approaches.

Informed by the literature, we contribute a theory-driven computational approach to measure inductive coding results, taking an important step towards addressing the broader concerns about efficiency and transparency in qualitative analysis. The next section will review existing ML/GAI work on this topic.

2.2 Machine-Assisted Qualitative Analysis

Many fields (e.g., computer vision, data analytics, etc.) have widely used ML/GAI methods to label data. Qualitative coding, whether inductive or deductive, can also mechanically be seen as applying labels (i.e., codes) to a given piece of data. This section reviews two ML/GAI perspectives for machine-assisted qualitative coding - *classification* and *generation*. While classification naturally fits into deductive coding scenarios with established evaluation metrics, its potential for inductive coding is intrinsically limited. Generation, on the other hand, has started to pick up traction for inductive coding. Yet, few studies have attempted to evaluate them, resulting in significant risks for research rigor.

2.2.1 ML/GAI For Classification. From a computational perspective, most current work views machine-assisted qualitative coding as a classification task, where algorithms aim to produce codes as similarly to human researchers as possible. More recently, researchers have worked on three classification approaches:

(1) **Supervised ML methods** that are trained on human-coded datasets. Liew et al., for example, approach machine-assisted qualitative coding as a "multi-label classification task" using SVMs trained on hand-coded datasets to aid in social science research. CoAICoder [31] similarly leveraged natural language understanding models trained on human coders' work to aid in collaborative qualitative coding.

(2) **Rule-based AI systems** that extract text-matching rules from human coding results and apply them to more datasets. For example, [55] and [33] encode human coders' coding results into rules for AI to suggest codes in unseen data. The usage of human-readable and editable rules increases the transparency in machine coding.

(3) **LLM-based methods** that instruct LLMs to label data with a predetermined codebook. For example, [71] used GPT-3 to label data with a fixed set of codes, thus supporting researchers in deductive coding. Despite the increased explainability, some evaluation studies have pointed to increased bias as a potential setback[3].

Classification approaches are generally easier to scale and evaluate, making large-scale datasets more accessible. Given the objective for algorithms to produce human-like codes, researchers have adopted traditional ML and deductive coding metrics to evaluate algorithm outcomes. For example, Liew et al. used human coding as the ground truth and applied two common ML metrics: precision, whether the positive predictions of the model were accurate, and recall, how well the model could identify positive instances in the training set. In Xiao et al., GPT-3's performance was evaluated with the inter-rater reliability (IRR) with human coders' independent coding results.

While the classification approach is more suitable for deductive coding, it has some intrinsic theoretical limitations for inductive coding. By positioning human coders' work as "ground truth," classification limits its outputs to labels

provided by human coders. While this limitation also applies to deductive coding, it is more pronounced in inductive coding, where coders are supposed to interpret "novel" insights from the dataset. They are expected to keep an open mind and capture as many codes as possible. As the process lacks a predefined "ground truth," truth-based (such as precision and recall) or convergence-based metrics, become much less useful (see 2.1.1).

2.2.2   ML/GAI For Generation. Open coding, on the other hand, is a natural fit for generation tasks: the number and nature of underlying codes (or themes) are unknown before the inductive process. While some scholars attempt to evaluate the results by matching human expert codes[51, 73], such evaluation underutilizes GAI's potential, which could aid researchers in identifying novel open codes for theory building. Two approaches have been more studied:

(1) **Topic modeling**, an unsupervised ML technique, identifies semantically similar word groups (topics) in text-based datasets. It has been used for inductive coding in various contexts, such as survey data[5], social media posts, and online discussions[57]. The resulting topics help researchers focus on key parts of the dataset and automate coding for future analysis. Despite recent efforts, the difficulty in interpreting and evaluating the results still limits its power[36, 60].

(2) **GAI** models have been more recently adopted for open, inductive coding. By iteratively providing data pieces with relevant instructions (e.g., research questions, coding instructions, desired output format), LLMs can produce codes that humans find more interpretable and useful [26, 61]. However, they may still miss nuance and struggle with less linguistically straightforward themes[26, 39], while generating non-grounded results or operating at a coarser level of analysis[16, 72]. Careful prompt design and prompting strategies are essential to mitigate these limitations, yet few comparison studies exist[16, 61].

Regardless of the method used, the lack of a "ground truth" and agreed-upon statistical metrics poses a greater challenge for evaluating open codes. While researchers can assess algorithms' output based on usefulness or explainability, the subjective measures are labor-intensive. Moreover, those measures may overlook the complexity of inductive coding. Even if all output codes seem useful or explainable, the algorithm may still miss critical codes without the evaluators' knowledge. Similar problems exist for De Paoli and Mathis's computational measurement, where the researcher sequentially feeds data pieces into gpt-3.5-turbo until no more codes are found: first, not all codes may uniformly exist among data pieces. If the LLM accidentally misses a code in the first interview, there is no guarantee that it will pick up again later; second, the LLM may be systematically biased to miss certain codes throughout the process.

As such, there is a pressing need to align theoretical expectations of qualitative analysis with practical evaluation mechanisms for machine-assisted inductive coding. Our work addresses this by introducing innovative, theory-informed computational methods for inductive coding and a novel computational approach for evaluating these results.

3   MEASURING INDUCTIVE CODES

Informed by literature (see 2.1.2), we developed a computational measurement that 1) aggregates open codes of multiple human/machine coders and 2) measures each against the aggregation. Using the method, we measured the outcomes of different ML/GAI approaches on two datasets and compared the first dataset's results with human evaluation. With human-AI collaboration, we explored the potential bias of ML/GAI coding approaches. Since part of our method involved generative AI, we validated its reliability with an output analysis.

3.1 The Conceptual Method

Following the suggestions of TA and GT, we adopt a team-based approach and measure individual coders' results against the aggregation of multiple coders. To achieve that, we proposed a conceptual structure, **Code Space (CSP)**, to represent inductive codes produced by each individual. The sum of individual CSPs becomes an **Aggregated Code Space (ACS)**. Using ACS as a reference for evaluation, we proposed four conceptual metrics (Coverage, Density, Novelty, and Divergence) to measure individual CSP's relative performance.
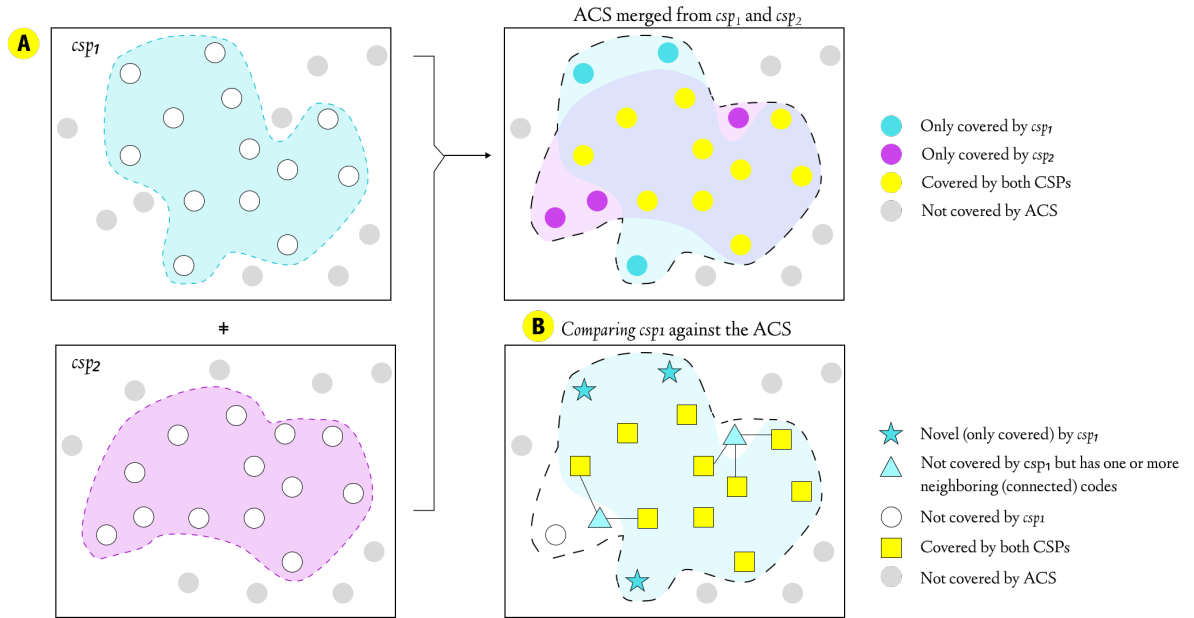

Fig. 1. **A**: A conceptual illustration of an ACS merged from csp_1 and csp_2. **B**: Measuring csp_1 using the merged ACS as a reference.

3.1.1 Code Spaces (CSP), Aggregated Code Spaces (ACS). Assuming we need to measure sets of inductive codes *Coding_Results*. We define a **Code Space** (CSP) as a multi-dimensional conceptual space that covers multiple codes identified or interpreted from the underlying qualitative dataset, each representing a set of codes. Each code is represented by a network node and has multiple dimensions related to its conceptual nature.

Combining multiple Code Spaces, we get a **Aggregated Code Space** (ACS) that covers all codes from all *Coding_Results* (Fig 1A). Similar but not equivalent codes are now connected by links and considered as *Neighbors*. The idea of ACS acknowledges that humans may not find all possible interpretations but could still build an "aggregated" set of codes to come closer. ACS is also a snapshot in time that documents researchers' current efforts. It can be used to look for convergence and divergence among the team members, supporting different stages of qualitative analysis. Convergence may help researchers determine the current consensus and indicate each code's visibility and importance. Divergence may reveal underlying biases, different focuses, or missed opportunities for researchers. For example, a recent study [45] critiques overreliance on consensus building and advocates for more attention to "dissonances, disagreements, and differences."

The concepts of CSP and ACS enable us to measure inductive coding results *Coding_Results*. While this paper may have coined the term ACS, many qualitative researchers have long relied on similar ideas and practices (e.g., Thomas). Examination of ACS could happen during open coding, when researchers discuss what was discovered and not discovered by individual team members to help them reorient the next coding batch; during axial coding, when researchers collaborate to consolidate the ACS into codebooks; and during selective coding, when researchers truncate the ACS to coalesce around a core category relevant to the research question.

3.1.2 Conceptualize the Evaluating Metrics. We propose four conceptual metrics to measure individual CSPs against their aggregation ACS (Fig 1B), which will be operationalized in later sections:

- **Coverage and Density**: How much conceptual space does a CSP cover, and how many codes does it use to cover this space? Both TA and GT strive for "richness" of codes. In practice, researchers need to strive for breadth and depth. Breadth means covering as diverse sets of concepts as possible. Depth means the descriptive details that ensures the concept's meaningfulness and richness, thus supporting researchers' further analysis [11, 23]. Combining them, researchers may be able to capture findings with depth and variation, two criteria for evaluating GT studies [23], to the extent that "nothing has been left out."[35] On the other hand, not all codes are of equal interest to researchers. Some concepts are more likely to be grounded in or relevant to the research question. The metrics must account for each concept's importance and weigh it accordingly.
- **Novelty**: How much of the "novel" conceptual space does a CSP include? We define "novel" codes as ones that fewer than *Novelty_Threshold* CSPs have included. Most of the time, CSPs will not be (even close to) identical. Sometimes, a researcher brings in novel ideas or lenses that were missed by others; in other cases, a researcher could make a slip or inappropriately name a code. Identifying "novel" codes could help us identify either scenario and support human-human or human-AI collaboration.
- **Divergence**: How far is a CSP's code distribution from the ACS? Suppose two CSPs, A and B, have the same coverage and density. However, A has more codes in common with most CSPs. B, on the other hand, has more codes in common with no one else ("novel" codes) or few in the group. To detect this at the macro level, we calculate each CSP's divergence as the separation from its probability distribution of codes to that of the ACS, using the latter as a "ground truth." In other words, the closer a CSP concept distribution is to the ACS, the less "divergent" it is.

Since our evaluation focuses more on the relative comparison between coding results, it does not directly address potential issues of groundedness, i.e., whether codes could be reasonably re-identified by another coder from the underlying data. However, our metrics do provide an indirect pathway for human-AI collaboration: by looking at the "novel" parts of CSP and/or codes with very few examples, we may be able to identify codes with potential groundedness issues more easily.

3.2 The Computational Method

The operationalization of our method starts from consolidating *Coding_Results* into an ACS for evaluation reference. Each *Coding_Result* should come from an individual machine or human coder. For *Coding_Result* to be treated as a CSP, each *Code* should have a *Label*, an optional list of *Definitions*, and a list of *Examples* (i.e. the pieces of data where the code was identified). In the output ACS, each *Code* will have a consolidated *Label*, *Definition*, a list of *Examples*, a list of *Neighbors* (other codes that are semantically close to this one), and a list of *Owners* (*Coding_Results* that have included one or more variations of the consolidated code). *Neighbors* enable the network structure of ACS

7

and the derivation of code *Clusters*. Evaluating individual CSPs against the ACS, we can calculate *Coverage*, *Density*, *Novelty*, and *Divergence* as overall and cluster-level metrics. For anonymization reasons, we will open-source our implementation after the paper's acceptance.

3.2.1 Consolidating the ACS. Overall, consolidating an ACS involves multiple iterations of:

(1) Finding equivalent codes and **merge** them into one.
(2) **Generate** a new label and definition for merged codes.
(3) **Repeat** the process with the new list of codes, until nothing more is merged.

Finding equivalent codes is more complicated than it seems. Human or machine coders often use different phrases to describe the same or similar ideas. For example, suppose coder A identified "user suggestion," while coder B found "user suggestions." They are clearly referring to the same idea. Suppose coder B found "user feedback" instead. Are A and B referring to equivalent, similar, or different idea(s)? To determine that:

(1) We encode each code with **text embedding**, transforming its *Label* and *Definition(s)* into a high-dimensional vector.
(2) We use **cosine distance**, a commonly used text similarity measure, to calculate the distance between codes.
(3) We use **a hierarchical clustering algorithm** to choose merging candidates, since it does not specify the expected number of results. For each node in the algorithm's dendrogram tree structure, we apply two input distance thresholds: *lower* and *upper*.
   (a) **Different codes**: Different codes with a distance above the *upper* threshold will never be merged.
   (b) **Equivalent codes**: Very similar codes with a distance below the *lower* threshold will always be merged.
   (c) For codes with a distance below the *upper* but above the *lower*, the algorithm penalizes[2] 1) the proportion of non-overlapping examples; and 2) the size of unique examples after merging (compared with the average example sizes of all codes).
      (i) **Also equivalent codes**: If the distance is below the *upper* after the penalty, the codes will be merged.
      (ii) **Similar codes**: If this is the last iteration, those codes become *Neighbors* to each other.

3.2.2 Implementing the Consolidation. We implemented the consolidating process by merging closer codes first and farther codes later. The purpose is to help LLMs create less general labels and definitions. We will discuss the choice of text embedding models and parameters in 3.3.3.

(1) We merge codes with exactly the same *Label*;
(2) We iteratively merge codes with very similar *Label* (*upper* = 0.35, *lower* = 0.35). Since codes in this step are usually very similar, to optimize the token usage, we simply use the shorter one. If the input does not include a *Definition*, we generate one for it.
(3) We iteratively merge codes with similar *Label* and *Definition* (*upper* = 0.5, *lower* = 0.4).
(4) We iteratively merge codes with similar *Label* and *Definition* (*upper* = 0.6, *lower* = 0.4). During the last iteration, we consider all codes with distances under *upper* to be *Neighbors*.

---

[2]The idea of **penalty** comes from our initial exploration, where merging by distance alone could lead to undesirable outcomes. For example, assume the code "user feedback" is close enough to "soliciting feedback" and "integrating feedback," yet the latter two are far enough apart. If an ACS starts with the latter two codes, they will not be merged. Yet, by adding "user feedback" to the ACS, a straightforward algorithm will merge all three codes into one, losing the nuances between "soliciting" and "integrating." To prevent this, our algorithm penalizes "oversized" merging results. For example, since "user feedback" will likely accumulate too many examples after merging the "soliciting feedback" and "integrating feedback," the distance threshold to merge into it will be stricter (lower). To reduce merging similarly named codes with different intentions, the algorithm also penalizes the differences between codes' examples.

Each iteration of the merging process is implemented as the follows:

(1) We calculate the penalty coefficient:

$$penalty = upper - lower \tag{1}$$

(2) For each pair of codes x, y, we calculate their distance with penalty:
  (a) We calculate **the cosine distance**:

$$distance(x, y) = 1 - \frac{x \cdot y}{\|x\| \, \|y\|} \tag{2}$$

  (b) Whenever $lower < distance(x, y) < upper$:
    (i) We calculate the percentage of the difference between examples of the two codes. Then, we squared the result to reduce the penalty on small differences.

$$diff(x, y) = (\frac{\#(examples[x] \cap examples[y])}{\#(examples[x] \cup examples[y])})^2 \tag{3}$$

    (ii) We apply the penalty of different examples on the distance.

$$distance(x, y) = distance(x, y) + diff(x, y) * penalty \tag{4}$$

(3) We use the **hierarchical clustering algorithm** to produce a dendogram tree structure from the distance matrix.
(4) We iterate through the tree structure from the top to the bottom. Each node of the tree represents a potential merge x, with a list of codes to merge; a $distance(x)$; and its unique examples after merged $examples(x)$.
  (a) We calculate the percentage of over-sizing (clamped between 0%-200%):

$$oversize(x) = clamp(\frac{\#(examples[x])}{avg(\#(examples))}, 100\%, 300\%) - 1 \tag{5}$$

  (b) We then calculate a new merging threshold with the penalty:

$$threshold(x) = upper - (oversize(x) * 0.5)^2 * penalty \tag{6}$$

  (c) We only merge the codes when $distance(x) < threshold(x)$.

3.2.3 Calculating the Metrics. We now calculate the metrics in 3.1.2 to measure individual CSPs against the consolidated ACS. To avoid CSPs with many redundant codes gaining an unfair advantage, whenever two or more codes are merged in the ACS, their CSP counterparts are also considered merged. Suppose coder A identified ten variations of the same concept "user suggestion", our method will treat them as only one code as long as they are detected and merged.

First, we calculate the *weight* of each code using **the sum of individual CSP's coverage** *value* **for this code**. The coverage *value* is 1 for CSP that has the code, or a percentage based on how many *neighbors* the CSP includes compared with the total number of *neighbors* of the code. For example, if two CSPs covered the same code and the third CSP covered two out of four neighbors but not the code itself, the *value* would be 1, 1, 0.5. The *weight* would be 2.5.

$$value(csp, code) = 1, if \; csp \in owners(code)$$
$$= \frac{\#(code \in ownedneighbors)}{\#(code \in neighbors)} \tag{7}$$

$$weight(code) = value(acs, code) = \sum_{csp \in acs} value(csp, code) \tag{8}$$

Then, we calculate the computational metrics weighted by the *value* of each code. For each CSP:

(1) **Coverage** measures the proportion of the CSP's total code *value* against the ACS's total code *weight*;

$$coverage\_value(csp) = \sum_{code \in acs} value(csp, code) \tag{9}$$

$$coverage(csp) = \frac{coverage\_value(csp)}{coverage\_value(acs)} \tag{10}$$

(2) **Density** measures the CSP's relative density (i.e. how many consolidated codes does the CSP include for its coverage) against the ACS;

$$density\_value(csp) = \frac{\#(code \in csp)}{coverage\_value(csp)} \tag{11}$$

$$density(csp) = \frac{density\_value(csp)}{density\_value(acs)} = \frac{\#(code \in csp)}{\#(code \in acs) * coverage\_value(csp)} \tag{12}$$

(3) **Novelty** measures the proportion of the CSP's total novel code *value* against the ACS's total novel code *weight*. In this study, we used $Novelty\_Threshold = 1$, i.e. a code is novel if only one codebook explicitly contains it;

$$novelty(csp) = \frac{\sum_{code \in csp (novel=1)} value(csp, code)}{\sum_{code \in acs (novel=1)} value(acs, code)} \tag{13}$$

(4) **Divergence** is calculated as the seperation between the CSP and the ACS's probability distribution. To measure that, we normalized the CSP and the ACS's values (or weights) into lists. Since individual CSPs may completely miss a code and its neighbors, we chose the Jenson-Shannon Divergence (JSD) to tolerate the resulting zero elements. We reported its metric version, Jenson-Shannon Distance, by taking a square root (citation here).

$$divergence(csp) = \sqrt{JSD(distribution(csp)||distribution(acs)} \tag{14}$$

ACS's **network structure** enables the detection of code clusters (or communities, **different from** the clustering algorithm we used for merging codes) and a more nuanced measurement. Since some codes may not have a neighbor, we also assign links from each code's three closest counterparts with a reduced weight for community detection with the Louvain algorithm[8]. For reproducibility, we report communities identified with seed = 0. We apply the metrics within each cluster to answer more nuanced questions, e.g., whether a human or machine coder could have oversampled or undersampled specific portions of codes.

3.3 Study Design

We conducted empirical experiments on two separate HCI datasets and research questions to demonstrate two use cases of our method. Since our measurement involves GAI in generating code labels and definitions, we validate its reliability by 1) comparing it with human evaluation results; and 2) analyzing its output stability.

3.3.1 Tasks and Datasets. We experimented with open coding results on two HCI datasets, each with its research question and context.

(1) Physics Lab's online community dataset (127 messages from the beginning of the community) between designers and teacher users. The research question was: "How did Physics Lab's online community emerge?" Four human

coders have previously open-coded the dataset. Three were PhD students, and one was an undergraduate assistant.

(2) Two interviews from a CHI 2024 study[20], each last around two hours. The research question was: "In the context of NetLogo learning and practice: What perceptions - strengths, weaknesses, and adoption plans - do interviewees perceive in LLM-driven interfaces? How do they use it to support their work? What are their needs for LLM-based interfaces?" Three human coders have previously open-coded the dataset. Two were PhD students, and one had a master's degree.

While both datasets consist of many more items, human experts spent tens of hours finishing open coding of the subset. On the other hand, since the goal is to identify concepts "as exhaustive as possible," the small dataset enables us to evaluate ML/GAI's potential in few-shot analyses.

3.3.2 Choice of ML/GAI Coding Approaches. In Case 1, we replicated five published ML/GAI approaches on open coding with both datasets[19]. We provided the same research question and dataset information for humans and machines. We assigned the same roles and tasks to machine coders, such as "You are an expert in thematic analysis with grounded theory, working on open coding." The exact prompts, parameters and Dataset 1's sample outputs can be found in[19], or supplementary materials of this paper. Here, we provide an overview of the five approaches:

(1) **BERTopic + LLM** uses topic modeling[68], an unsupervised ML technique to identify groups of semantically similar words (i.e., topics) and explains the topics with LLMs. Since BERTopic's instruction was intended for general-purpose label-making, we gave additional instructions about the research question and contexts.

(2) **Chunk Level** asks LLMs to identify open codes from chunks (e.g., a conversation, an interview, etc.) of data. Many variants of this approach have been adopted by recent papers (e.g., [41, 67]).

(3) **Chunk Level Structured** asks LLMs to generate more than one level of concepts: the first for "categories" or "themes," the second for "codes" or "subcodes." Some recent papers have started to adopt this approach[19, 61].

(4) **Item Level** asks LLMs to conduct line-by-line coding, as suggested by the grounded theory literature[34]. A few papers have adopted this approach to generate one [61] or multiple codes [19] per line.

(5) **Item Level with Verb Phrases** builds on the previous approach but instructs LLMs to use verb phrases for labels explicitly. The design was inspired by a grounded theory literature[25] and reported by a recent study[19].

Table 1 presents an excerpt from Dataset 1, coded by four human and five machine approaches. All machine coders were driven by the same model (GPT-4o-0513) with the same temperature (0.5).

3.3.3 Choice of Models and Hyperparameters. Our computational method utilizes text embedding and generation models during evaluation. For text embedding models, we consulted the MTEB leaderboard between May and June 2024 and experimented with a few alternatives. For this study, we used gecko-768 from Google due to its relatively high performance, low dimensions (with better computational efficiency), and easy accessibility.

We used cosine distance for thresholds in 3.2.2, where 0 means the two codes are identical; 1 for no correlation; and 2 for absolutely different. During our initial experiments, we set the thresholds of $lower = 0.4$ and $upper = 0.6$ based on two indicators: 1) we referred to the distribution of pairwise distances and located local turning points; and 2) whether the code pairs under the criteria meet our definition, where human can recognize codes closer than $lower$ as the same idea, while codes further than $upper$ are likely to be different. Different research contexts and text embedding models may require different parameters.

11

User 4232	If there could be an export function, or the ability to save or import experiments, it would be convenient. We could set up the parameters in the office and directly import them in class.
BERTopic + LLM	feature requests for physics experiments
Chunk Level	community feedback loop, context of use, participatory design, user feedback and suggestions
Chunk Structured	user feedback, practical application, usability improvements, feature requests
Item Level	feature request, usability improvement, classroom application, import/export functionality
Item Verb Phrases	propose additional features, emphasize convenience, suggest practical use case
4 Human Coders	community feedback, feature request, propose another feature, gives reason for it
Designer	There will be.
BERTopic + LLM	future planning and development
Chunk Level	designer responsiveness
Chunk Structured	designer response, acknowledgment and implementation
Item Level	designer confirmation, feature implementation, future planning
Item Verb Phrases	confirm future implementation, validate user request, plan feature development
4 Human Coders	developer response, acknowledgement, acceptance of request, promising it will realize
Designer	Does the class have internet?
BERTopic + LLM	technical and infrastructural challenges in educational settings
Chunk Level	contextual constraints
Chunk Structured	feature updates and enhancements, user-designer interaction
Item Level	internet availability inquiry, classroom setup, technical consideration
Item Verb Phrases	inquire about classroom conditions, gather context, consider technical requirements
4 Human Coders	engaging with community, seeking context, ask followup question on usage scenario
User 4232	Generally not. Ever since an adult image popped up during a major city-level open class, the school has disabled the network on classroom computers [Emoji].
BERTopic + LLM	technical and infrastructural challenges in educational settings
Chunk Level	context of use, contextual constraints
Chunk Structured	feature updates and enhancements, user-designer interaction
Item Level	internet restriction, classroom environment, security concern, past incident
Item Verb Phrases	explain lack of internet, provide context, share past incident
4 Human Coders	contextualizing response, humor, personal anecdote, sharing information for design, story sharing, gives an answer, explains the answer

Table 1. An example exchange between a teacher user and a designer from our Dataset 1, coded by five machine coders and four human coders. We lightly merged very similar codes (e.g., ask a question vs. question) from the four human coders to save space.

To understand how the choice of model and temperature may influence the **evaluation** process, we used GPT-4o-0513 (0513), GPT-4o-mini, and Llama3-70B because: 1) GPT-4o-0513 was one of the most powerful models (in terms of state-of-the-art evaluation benchmarks) at the time of the experiment; 2) GPT-4o-mini was one of the most potent smaller models; 3) Llama3-70B was one of the most potent open-source models.

3.3.4 Experiment Design. We conducted two case studies to understand two research questions:

- What can we learn about existing ML/GAI approaches for open coding?
- Is our method statistically **reliable** enough to evaluate open codes from **machine and human coders**?

Case 1 examines the five machine coding approaches using 1) the overall metrics for machine and human coders; and 2) the cluster-level metrics of relative coverage, a metric between -100% (completely undersampled) to +inf (extremely oversampled). Two human researchers independently interpreted the theme for each network cluster (see 3.2.3) based on its constituent codes and reconciled their differences into a single label.

$$coverage_{relative}(csp, cluster) = \frac{coverage(csp, cluster)}{coverage(csp)} - 100\% \tag{15}$$

Case 2 evaluated six mainstream LLMs' performance in open coding with the same prompt: GPT-3.5-turbo; GPT-4o-0513; Llama3-70B; Mixtral 8x22b; Claude3-haiku; and Claude3.5-sonnet. [3] To save space, we only report the metrics from Item-level Verb Phrases, the best-performing GAI approach. The measurement includes all machine and human coders' results, enabling comparisons between LLMs; and between the aggregated results of LLMs and humans.

The **reliability study** evaluated our measurement's outcome stability based on Case 1. We repeated the measurement with three models, five temperatures (0, 0.25, 0.5, 0.75, 1). This results in 15 combinations of LLM and temperature. For each combination, we repeated 10 independent runs, totaling 150 runs. We recorded the evaluator LLM, temperature, run number, coder identity, and four metrics. With those, we evaluated:

- The **coefficient of variation** *cov* for each metric, each CSP, and each combination of LLM and temperature. On each dataset, we ran fixed-effects regressions to understand if the choice of LLM, temperature, or CSP's *mean* value of metric influenced the *cov*.
- **Whether using different combination of LLM and temperature impacts pairwise comparisons**. On each dataset, we ran a series of pairwise ANOVA experiments and recorded significant pairs on 1) the entire output from 150 runs; 2) each of the 15 combinations. We calculated the intersection of all pairs and looked for any individual combination that would produce different conclusions than the entirety. For example, suppose we found the Item-Level approach's coverage significantly higher than the Chunk-Level approach. If we only evaluated with one LLM and one temperature, will the conclusion change?

Since we found the choice of LLM and temperature to have no significant impact on the pairwise comparison outcome, we only conducted 10 evaluation runs with Llama 3-70B at 0.5 temperature for Case 2.

3.4 Empirical Results

3.4.1 Case 1: Evaluating Machine Coding Approaches. Figure 2, 3 demonstrates *coverage* and *divergence*, two of the more stable metrics (see 3.4.3), for all machine and human coders in both datasets. We noted:

- Similar to a previous human evaluation[19], we found that item-level coding approaches have higher coverage and lower divergence than chunk-level and BERTopic approaches.
- While the aggregation of human researchers has higher coverage and lower divergence than any single machine coding approach in coverage and most in divergence, the aggregation of machine coders has higher coverage and lower divergence than each and all human researchers. Note that we do not claim that machines outperformed humans, and an ongoing study is currently examining the implications.
- Note that the numbers of codes do not strictly correlate with the coverage metric. For example, in Dataset 1, the Item-Level Verb-Phrase approach found 282 raw codes with 79.1% coverage, while the aggregation of human coders found 340 raw codes with only 75.5%.

We further explored the potential biases of human and machine coders with cluster-level metrics. Tables 2, 3 presents clusters for both datasets' ACS and the relative coverage of each CSP. Clusters are sorted by the sum of their component codes' weights. Thus, clusters with more codes and higher consensus levels are listed first. Numbers higher than

---

[3]The list does not include GPT-4o-mini, released after we completed the coding task.
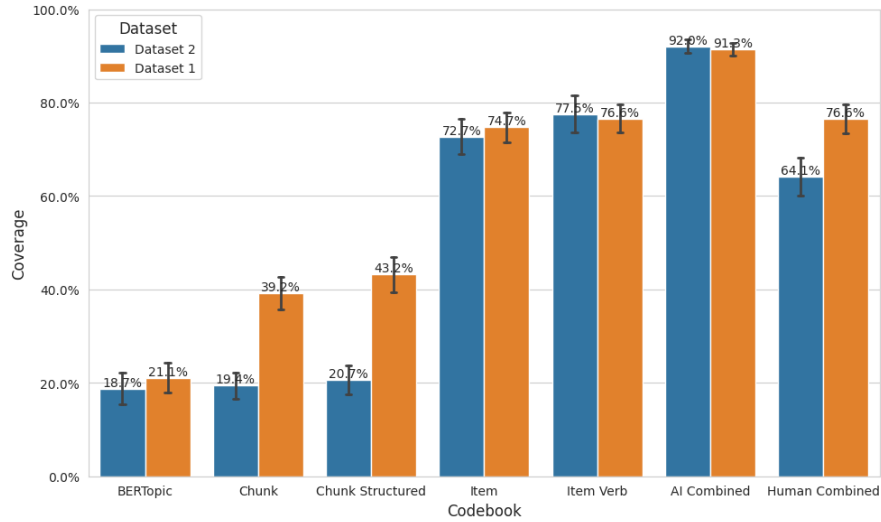
Fig. 2. The *coverage* metric for all coding approaches, Dataset 1 (with four humans) and 2 (with three humans). **For grounded theory open coding, higher is better.**
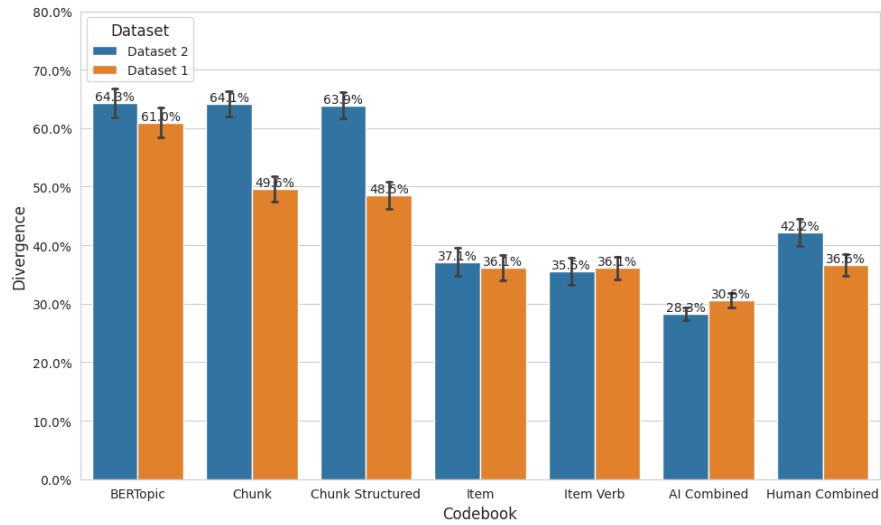

Fig. 3. The *divergence* metric for all coding approaches, Dataset 1 (with four humans) and 2 (with three humans).

0 represent oversampling of codes, while numbers lower than 0 represent undersampling. The standard deviation indicates the degree of imbalance between clusters.

We immediately noted that almost all machine and human coders had completely missed some clusters from the data. Across the two datasets, the only exception is Item-level LLM coding with verb phrases. It also has the lowest *std* of relative coverage across all individual coders, showing its comparatively uniform sampling across all code clusters comparable with the aggregation of all human coders. Examining the qualitative nature of missed or undersampled

ID	#	BERTopic	Chunk	Chunk-S	Item	Item-V	All AI	All Human
1	34	+2.30%	+9.90%	+13.80%	+26.40%	+12.00%	+6.60%	+14.40%
2	45	+99.10%	+16.00%	-43.10%	+12.10%	-1.50%	+3.80%	-12.20%
3	26	+73.10%	+59.20%	+58.20%	+15.20%	+15.20%	+3.30%	+25.80%
4	28	+1.40%	+62.80%	+46.00%	-5.00%	+13.60%	+2.20%	+12.20%
5	26	-75.00%	+29.50%	+44.90%	-17.20%	+3.10%	-3.80%	+8.30%
6	35	-86.20%	-65.80%	-65.90%	-19.90%	-40.50%	-13.50%	-2.70%
7	20	+12.40%	+36.50%	+17.50%	+28.10%	+13.40%	+9.30%	-6.50%
8	23	-14.20%	-80.50%	-37.90%	-1.80%	+0.00%	-4.50%	-6.30%
9	20	+1.80%	-55.00%	-19.30%	+12.50%	-14.80%	-1.10%	-19.40%
10	16	+20.20%	+31.90%	+41.70%	-1.00%	-9.60%	+5.40%	-19.70%
11	16	-100.00%	-70.40%	+30.30%	-40.70%	+10.10%	-0.50%	+5.10%
12	16	+116.10%	-20.40%	-100.00%	-42.40%	-27.10%	-18.90%	-8.30%
13	11	-100.00%	-48.30%	-7.10%	-2.00%	+11.30%	+6.60%	+5.90%
14	14	-100.00%	-100.00%	-100.00%	-54.60%	-32.60%	-3.80%	-64.70%
15	10	-100.00%	-100.00%	-80.70%	-54.60%	-5.20%	+1.70%	-55.90%
16	10	-100.00%	-100.00%	-100.00%	-50.50%	-42.50%	-29.40%	-51.90%
17	6	-100.00%	-100.00%	-100.00%	-100.00%	-57.90%	-63.00%	-11.70%
Std		75.26%	60.30%	59.03%	34.98%	23.10%	17.81%	25.37%

Table 2. ACS of Dataset 1: Clusters of Codes, and Each Coding Approach's Relative Coverage (-100% = Completely Missed; +Inf = Infinitely Oversampled).

clusters is crucial to understanding potential bias further. It allows researchers to situate the analysis in the context of the data and research questions. Below, we briefly interpret the clusters missed by each approach:

- **BERTopic + LLM**. The coverage of BERTopic is low (21.1%) and highly unevenly distributed among clusters. The seemingly oversampling on cluster 12 (designers' reactions to user suggestions) was only because of the low baseline number, as it only included two (plus one neighbor) out of 14 codes. Moreover, BERTopic misses clusters 13-17, where other coders noted many details about the design exchanges, such as needs, design considerations, and communication strategies.
- **Chunk-level approaches**. Both approaches have relatively low coverage (39.2%, 43.2%). The structured approach has slightly better coverage, but the overall unevenness level within clusters stays similar, in addition to completely missing cluster 12. Both start to cover cluster 13 (designers' apologies, explanations, and reassurance to users), yet still miss clusters 14-17. Both approaches also oversample positive expressions (e.g., cluster 3) while undersampling clusters with negative implications (e.g., clusters 6 and 12).
- **Item-level approaches**. Both approaches have high coverage (74.7%, 76.6%). The verb phrase approach has slightly better coverage and unevenness level and avoids the baseline approach's miss of cluster 17 (designers' response strategies). This is not a small achievement: all other coders except one human missed this cluster of 6 codes altogether.

We observed similar trends in Dataset 2, where BERTopic, chunk-level LLM coding, and human coders missed many clusters around interviewees' nuanced reflections (clusters 12-19). Moreover, item-level coding approaches continued to do better than other approaches. Notably, human coders miss cluster 19 (reflecting and criticizing working and help-seeking cultures) altogether.

ID	#	BERTopic	Chunk	Chunk-S	Item	Item-V	All AI	All Human
1	21	-30.10%	+8.10%	-48.50%	+18.60%	+14.90%	+7.60%	+13.00%
2	26	-53.00%	+68.50%	+39.50%	+1.70%	+13.90%	-0.40%	+23.70%
3	24	+83.00%	-22.80%	+43.70%	+8.50%	-3.00%	-2.40%	+39.10%
4	22	+151.50%	-62.40%	+104.80%	+13.00%	+4.20%	+0.30%	+16.30%
5	20	+134.40%	-36.00%	+96.10%	-7.00%	+4.10%	-2.80%	+14.00%
6	15	+86.20%	+124.90%	-38.80%	+16.60%	-10.20%	+4.70%	-11.50%
7	15	-17.50%	-100.00%	-63.00%	+16.50%	+3.00%	+4.70%	+22.20%
8	16	-55.80%	+54.10%	-100.00%	+1.70%	+5.90%	-4.40%	+12.60%
9	17	-81.70%	+129.60%	+56.30%	-14.40%	-10.90%	+0.20%	-14.00%
10	16	-100.00%	+49.40%	+1.70%	-9.50%	+4.10%	+3.40%	-52.00%
11	12	-36.40%	+55.30%	+5.70%	-11.10%	-1.60%	-5.30%	-6.40%
12	14	-100.00%	-100.00%	-48.50%	-10.90%	-16.10%	-2.20%	-36.20%
13	14	-100.00%	-100.00%	-48.50%	-10.90%	+11.80%	+7.60%	-85.80%
14	16	-100.00%	+75.20%	-100.00%	-51.80%	-54.70%	-15.10%	-42.50%
15	10	-100.00%	-100.00%	-100.00%	-4.20%	-1.60%	+7.60%	-58.40%
16	7	-41.10%	-100.00%	-100.00%	-27.40%	-18.00%	+7.60%	-48.00%
17	7	-100.00%	-100.00%	-100.00%	-81.30%	-47.30%	-38.50%	-33.20%
18	5	-100.00%	-100.00%	-46.00%	-44.00%	-29.70%	-23.20%	+11.40%
19	5	-100.00%	-100.00%	-46.00%	-6.60%	-12.10%	+7.60%	-100.00%
Std		87.17%	87.28%	68.77%	25.24%	18.97%	12.03%	39.81%

Table 3. ACS of Dataset 2: Clusters of Codes, and Each Coding Approach's Relative Coverage (-100% = Completely Missed; +Inf = Infinitely Oversampled).

3.4.2 Case 2: Benchmarking Mainstream LLMs. In Figures 4 and 5, we applied our method to benchmark six mainstream LLMs in mid-2024. We noted:

- **Significant gaps** in different models' coverage and divergence. In both datasets, we found GPT-4o, Claude 3.5 Sonnet, and Llama3-70B significantly outperformed GPT-3.5-Turbo, Claude 3 Haiku, and Mixtral 8x22B. Without designating a ground truth, our methods roughly distinguish larger, more "powerful" models (measured by common ML benchmarks, e.g., MMLU) from smaller ones.
- Machine coders' performances were **not always consistent** among the two datasets. For example, all models but Claude 3 Haiku have seen a performance drop in Dataset 2, with Claude 3.5 Sonnet dropping almost 12%.
- Machine coders' aggregated result consistently **achieved better coverage, density, and divergence** than human coders' aggregated results, with coverages around 95% and significantly lower divergences in both datasets. Even when most individual models' performance dropped in Dataset 2, the performance of machine coders' aggregation almost stayed the same.
- We also noted the performance of aggregated human coders dropping in Dataset 2. One potential reason: only 3 human coders conducted open coding for Dataset 2, compared with 4 for Dataset 2.

3.4.3 Reliability Study. Figures 6, 7 demonstrate the outcome variance of each metric across three LLMs on two datasets. We noted:

- Overall, *divergence* consistently has the lowest standard variance (*std*, 2-3%) and coefficient of variance (*cov*, 1.5-5.5%), while *novelty* has the highest *std* (3-6%) and *cov* (8-65%).
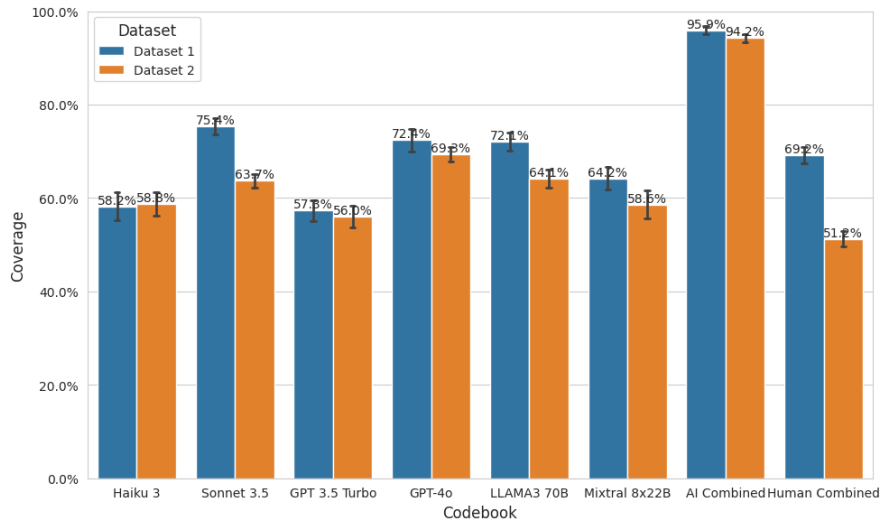
Fig. 4. The *coverage* metric for six LLMs, Dataset 1 (with four humans) and 2 (with three humans). **For grounded theory open coding, higher is better.**
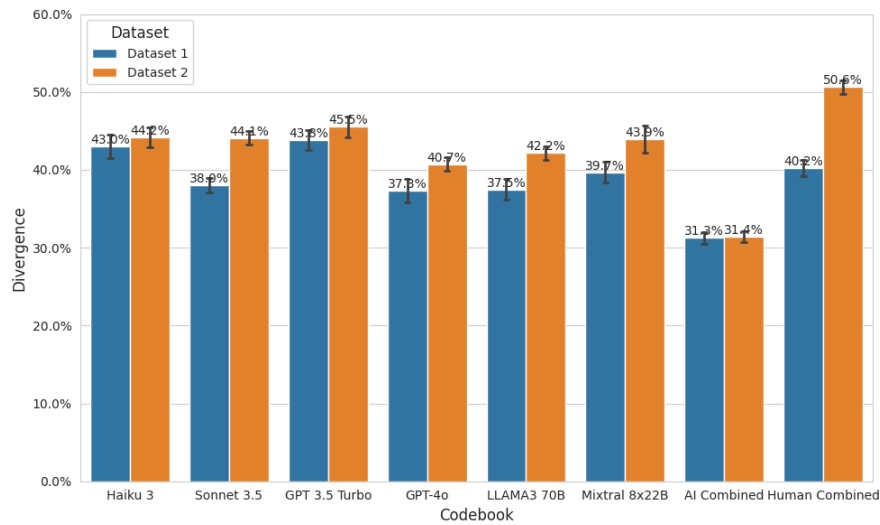

Fig. 5. The *divergence* metric for six LLMs, Dataset 1 (with four humans) and 2 (with three humans).

- After controlling the temperature and coder identities, the model choices **occasionally have a significant but small** impact. However, there is no relationship between the sizes of models and the *cov* impacts, and **no models have consistently lower** *cov*.
- The model temperature **does not** have a significant impact.
- For *coverage*, *density*, and *novelty*, their *cov* are negatively correlated with individual coders' *mean* value of the metric. For example, a coder with higher *coverage* will likely have a lower *cov* than a coder with lower *coverage*.
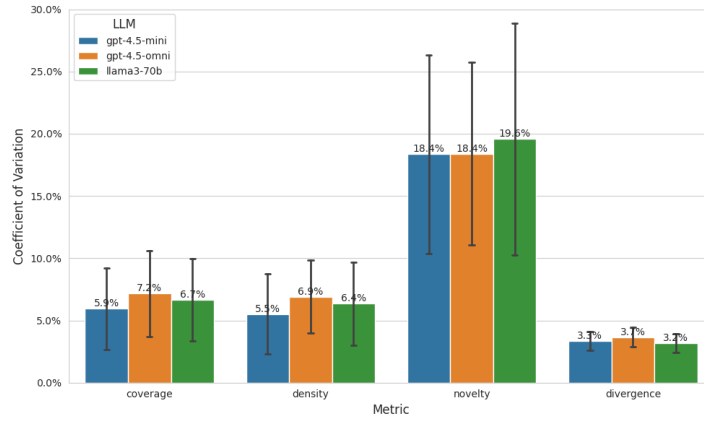
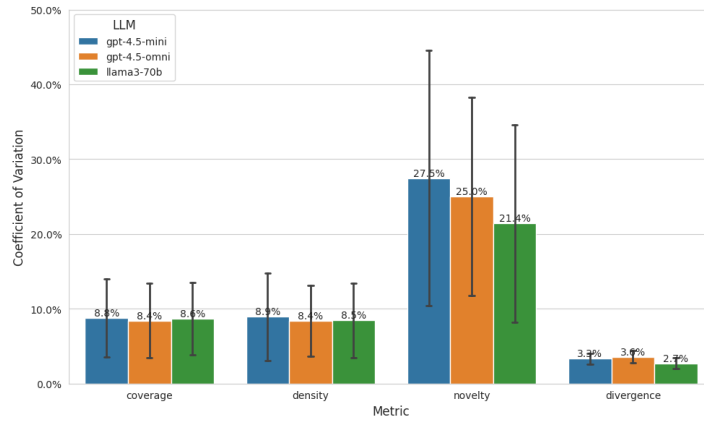Fig. 6. Coefficient of Variance for Each Metric among LLMs, Dataset 1

Fig. 7. Coefficient of Variance for Each Metric among LLMs, Dataset 2

Since running 150 evaluation runs for a single comparison study may be cost-ineffective, we evaluated our method's reliability when only one model, and temperature, and 10 evaluation runs are used. Here, we assume the pairwise comparison results of 150 runs as the ground truth. Across all 15 combinations on four metrics and two datasets, we only identified two cases of **false positives** for a pair with a minuscule mean difference of *density* (BERTopic vs. Chunk-level Structured, 32.71% vs. 32.41%) in Dataset 1. Besides that, we only identified **false negatives** for comparison pairs with small mean differences. In other words, unless the goal is to compare metrics with small numerical differences, 10 runs would be generally sufficient.

3.5 Discussions

3.5.1 Reliability of the Measurement. While using GAI in our method inevitably introduces stochasticity and potential biases, we established the reliability of our computational measurement with the reliability study (3.4.3).

Our method's only source of stochasticity comes from GAI's generation of code definitions and labels, which influences the downstream decision of code merging. Naturally, coding results with fewer codes will be impacted more, and our empirical results support this assumption. In plain words, our method provides more stable outcomes at evaluating "better" coding results but less so for "worse" ones. This is especially true for *novelty*, which measures the number of codes that *no one else has identified*. Therefore, we only recommend using *novelty* as a qualitative indicator for potential outliers, while using *divergence*, *coverage*, and *density* for quantitative comparisons.

We confirmed that using a single LLM and temperature does not generally lead to false reports on paired comparisons. In only one case, 10 runs with a single LLM and temperature caused a false positive against the entirety of 150 runs. Moreover, the result shows that our method does not rely on the most powerful models. Open-source models with the potential for local deployment and better privacy protection (such as Llama3-70B) or smaller models with cheaper costs and less ecological footprints (such as GPT-4o-mini) will likely work for most scenarios.

3.5.2 Suggestions for Using ML/GAI in Inductive Coding. Case 1 reaches similar conclusions as a previous human evaluation[19]: the item-level approaches performed best on both datasets and research questions. The cluster-level metrics enabled us to compare ML/GAI coding approaches with more nuances. BERTopic and chunk-level approaches had worse overall metrics, missed entire clusters of codes, and were less capable of identifying **nuances** of human interactions, particularly ones with **seemingly** less connection with the research question. They also oversampled positive emotions or feedback, implying a potential bias for their coding results.

In contrast, the success of item-level approaches suggests the potential of embedding human processes for qualitative analysis into LLM prompts. While we started mentioning grounded theory in the Chunk-Level Structured approach, its performance was not much better than the Chunk-Level. The situation changed when we instructed LLMs to strictly follow the grounded theory process[62]: by coding the data item-by-item, the item-level approaches produced much better results. Following existing literature[25], our adoption of verb phrases as labels further enabled a more nuanced interpretation of the data.

Case 2 provides a quick assessment of state-of-the-art LLMs available in mid-2024. Our finding suggests that existing NLU benchmarks for LLMs may positively correlate with their performance on inductive qualitative coding. On the other hand, the same LLMs may perform differently on different research questions and datasets, necessitating broader evaluative studies. While we suggest researchers use the best available model, our findings also indicate the feasibility of powerful open-source models such as Llama-3 70B, particularly for scenarios where data privacy concerns are stronger. Moreover, we suggest using multiple models simultaneously as an even better approach. The combination of six models consistently covered almost all codes identified by human researchers, even when individual performances varied across datasets.

3.5.3 Human-AI Collaboration in Interpretation. This work's core contribution is a novel computational measurement to understand, compare, and evaluate open codes from multiple machine or human coders. Hence, it is crucial to share how we interpreted the algorithmic results and discuss scenarios where our measurement could contribute to human-AI collaboration in open coding processes.

The nature of qualitative research prevents us from making a general claim with only two datasets and research questions, yet this is more of a feature than a bug. Both the open codes and the corresponding evaluation are naturally bounded by the datasets, research questions, and perspectives researchers adopted. Thus, results from our measurement must be interpreted in context. Below, we present an example flow based on Table 2 on Dataset 1:

- A researcher first looked at the general metrics of each CSP in a random evaluation run of Case 1, where BERTopic had the highest *divergence* across all runs. Since the researcher believes the ACS to be reasonably well-covered due to the presence of multiple human coders, it seems that BERTopic produced the most deviant results.
- Here, two possibilities exist: either BERTopic identified something novel and insightful, or it simply missed too many codes. To evaluate this, the researcher checked the *novelty* metric and found BERTopic the lowest again. So it is unlikely to be the first situation; in fact, it only has 3 codes that no one else found. Those codes are all related to aspects of the software feature, which is far from the research question about community formation.
- Note that BERTopic only identified 23 codes. For low-level coding approaches that found hundreds of codes, the researcher may need additional support to understand their potential biases. Therefore, the researcher clustered codes in the ACS and calculated cluster-level metrics.
- The researcher found that the Item-Level approach misses and under-samples more clusters than the Item-Level Verb Phrase approach, indicating a higher risk for potential bias. From there, based on the research goals and questions, the researcher qualitatively interpreted the two approaches' outcome differences and decided to adopt the Item-Level Verb Phrase approach for the rest of the coding process.

While computational measurement is crucial in the process, such interpretations are only possible through a qualitative understanding of the codes and clusters. This paper's human evaluation focuses on the open coding processes, where every grounded interpretation is welcomed. However, when researchers conduct further analysis steps, depending on the research context, a code could become too nuanced and with too few cases; too broad and covers everything; irrelevant to the core aspects of the research question; or has been reported many times before. While our convergence metrics automatically devalue outlier codes from few coders, it is still possible that a few "well-performed" approaches or models actually contribute little to the final analysis. It still relies on humans (potentially, in collaboration with AI) to make these interpretations and decisions, opening up opportunities for further research work and interface design.

The potential of our evaluation method is not limited to machine coders alone. In Case 1, we presented how individual human coders may have missed insights from the data. During our study process, we found the visualization of ACS's network structure particularly useful, as it revealed the different focuses of human coders and facilitated fruitful discussion around the research question. Moving forward, we will focus on design work that can materialize the method's potential in supporting human coders, with or without machine coders in ACS.

ACKNOWLEDGMENTS

REFERENCES

[1] Anne Adams, Ann Blandford, and Peter Lunt. 2005. Social empowerment and exclusion: A case study on digital libraries. *ACM Transactions on Computer-Human Interaction* 12, 2 (June 2005), 174–200. https://doi.org/10.1145/1067860.1067863

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

[3] Julian Ashwin, Aditya Chhabra, and Vijayendra Rao. 2023. Using large language models for qualitative analysis can introduce serious bias. *arXiv preprint arXiv:2309.17147* (2023).

[4] Jennifer Attride-Stirling. 2001. Thematic networks: an analytic tool for qualitative research. *Qualitative Research* 1, 3 (Dec. 2001), 385–405. https://doi.org/10.1177/146879410100100307

[5] Eric P. S. Baumer, David Mimno, Shion Guha, Emily Quan, and Geri K. Gay. 2017. Comparing grounded theory and topic modeling: Extreme divergence or unlikely convergence? *Journal of the Association for Information Science and Technology* 68, 6 (2017), 1397–1410. https://doi.org/10.1002/asi.23786 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.23786.

[6] Izhak Berkovich. 2018. Beyond qualitative/quantitative structuralism: The positivist qualitative research and the paradigmatic disclaimer. *Quality & Quantity* 52, 5 (2018), 2063–2077. https://idp.springer.com/authorize/casa?redirect_uri=https://link.springer.com/article/10.1007/s11135-017-0607-3&casa_token=JlBYcYvjqpwAAAAA:Qn6arDr_zncTUy62N2jsyRQgPblLU2nVHRlN6nUkQib4UkbPw4-HgrsOhMi0Bhsz-lFEhcU2DbSDb58o Publisher: Springer.

[7] Andrea Bingham and Patricia Witkowsky. 2021. Deductive and Inductive Approaches to Qualitative Data Analysis. In *Analyzing and Interpreting Qualitative Research: After the Interview.* SAGE Publications, Inc. https://books.google.com/books?hl=en&lr=&id=0xIoEAAAQBAJ&oi=fnd&pg=PA133&dq=deductive+coding+qualitative&ots=1wZ007ufMy&sig=tLe5PscNzbMG_yMvXduwFPOquvY#v=onepage&q=deductive%20coding%20qualitative&f=false

[8] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[9] 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.* ACM, Hamburg Germany, 1–18. https://doi.org/10.1145/3544548.3581203

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (Jan. 2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[11] Virginia Braun and Victoria Clarke. 2013. Successful qualitative research: A practical guide for beginners. (2013).

[12] 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 (July 2021), 328–352. https://doi.org/10.1080/14780887.2020.1769238

[13] Virginia Braun and Victoria Clarke. 2022. Conceptual and design thinking for thematic analysis. *Qualitative Psychology* 9, 1 (Feb. 2022), 3–26. https://doi.org/10.1037/qup0000196

[14] 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 (Aug. 2004), 247–265. https://doi.org/10.1177/1468794104044434

[15] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. 2021. To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *Proceedings of the ACM on Human-computer Interaction* 5, CSCW1 (2021), 1–21.

[16] Courtni Byun, Piper Vasicek, and Kevin Seppi. 2024. Chain of Thought Prompting for Large Language Model-driven Qualitative Analysis. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24).* Association for Computing Machinery.

[17] Lara Carminati. 2018. Generalizability in Qualitative Research: A Tale of Two Traditions. *Qualitative Health Research* 28, 13 (Nov. 2018), 2094–2101. https://doi.org/10.1177/1049732318788379

[18] 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 (May 2019), 116–130. https://doi.org/10.1177/1525822X19838237

[19] John Chen, Alexandros Lotsos, Lexie Zhao, Grace Wang, Uri Wilensky, Bruce Sherin, and Michael Horn. 2024. Prompts Matter: Comparing ML/GAI Approaches for Generating Inductive Qualitative Coding Results. arXiv:2411.06316 [cs.CL] https://arxiv.org/abs/2411.06316

[20] John Chen, Xi Lu, Yuzhou Du, Michael Rejtig, Ruth Bagley, Michael S. Horn, and Uri J. Wilensky. 2024. Learning Programming of Agent-based Modeling with LLM Companions: Experiences of Novices and Experts Using ChatGPT & NetLogo Chat. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems.*

[21] Ned Cooper, Tiffanie Horne, Gillian Hayes, Courtney Heldreth, Michal Lahav, Jess Scon Holbrook, and Lauren Wilcox. 2022. A Systematic Review and Thematic Analysis of Community-Collaborative Approaches to Computing Research. In *CHI Conference on Human Factors in Computing Systems.* 1–18. https://doi.org/10.1145/3491102.3517716 arXiv:2207.04171 [cs].

[22] Juliet Corbin and Anselm Strauss. 2008. 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. https://doi.org/10.4135/9781452230153

[23] Juliet Corbin and Anselm Strauss. 2008. 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. https://doi.org/10.4135/9781452230153

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

[25] Natalie R. Davis, Shirin Vossoughi, and John F. Smith. 2020. Learning from below: A micro-ethnographic account of children's self-determination as sociopolitical and intellectual action. *Learning, Culture and Social Interaction* 24 (2020), 100373. https://www.sciencedirect.com/science/

article/pii/S2210656119302107?casa_token=R60cWsxYBRgAAAAA:JTuwj92trq-YVKvmrCwtjFxlRCgBsFxzdpnKu2WNMa9l2OqmQ8wO_h_qvLqL49FXXQAx1gXPqw Publisher: Elsevier.

[26] Stefano De Paoli. 2023. 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 (Dec. 2023), 1–23. https://doi.org/10.1177/08944393231220483

[27] Stefano De Paoli and Walter S Mathis. 2024. Reflections on inductive thematic saturation as a potential metric for measuring the validity of an inductive thematic analysis with LLMs. *Quality & Quantity* (2024), 1–27.

[28] Jennifer Fereday and Eimear Muir-Cochrane. 2006. Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods* 5, 1 (March 2006), 80–92. https://doi.org/10.1177/160940690600500107

[29] Nick J Fox. 2008. Post-positivism. *The SAGE encyclopedia of qualitative research methods* 2, 1 (2008), 659–664.

[30] 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*. ACM, Vancouver BC Canada, 113–122. https://doi.org/10.1145/1978942.1978960

[31] Jie Gao, Kenny Tsu Wei Choo, Junming Cao, Roy Ka-Wei Lee, and Simon Perrault. 2023. CoAIcoder: Examining the Effectiveness of AI-assisted Human-to-Human Collaboration in Qualitative Analysis. *ACM Transactions on Computer-Human Interaction* 31, 1 (Nov. 2023), 6:1–6:38. https://doi.org/10.1145/3617362

[32] Susan Gasson. [n. d.]. Rigor In Grounded Theory Research: An Interpretive Perspective on Generating Theory From Qualitative Field Studies. ([n. d.]).

[33] 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)*. Association for Computing Machinery, New York, NY, USA, 1–19. https://doi.org/10.1145/3544548.3581352

[34] Graham R. Gibbs. 2007. Thematic coding and categorizing. *Analyzing qualitative data* 703, 38-56 (2007). https://study.sagepub.com/sites/default/files/analyzing-qualitative-da.pdf

[35] Barney G Glaser, Judith Holton, et al. 2004. Remodeling grounded theory. In *Forum qualitative sozialforschung/forum: qualitative social research*, Vol. 5.

[36] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794* (2022).

[37] Egon G Guba and Yvonn A S Lincoln. 1994. Competing Paradigms in Qualitative Research. In *Handbook of qualitative research*. Sage Publications, Inc., 105–117.

[38] Greg Guest, Emily Namey, and Mario Chen. 2020. A simple method to assess and report thematic saturation in qualitative research. *PloS one* 15, 5 (2020), e0232076. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0232076 Publisher: Public Library of Science San Francisco, CA USA.

[39] Leah Hamilton, Desha Elliott, Aaron Quick, Simone Smith, and Victoria Choplin. 2023. Exploring the Use of AI in Qualitative Analysis: A Comparative Study of Guaranteed Income Data. *International Journal of Qualitative Methods* 22 (March 2023), 16094069231201504. https://doi.org/10.1177/16094069231201504 Publisher: SAGE Publications Inc.

[40] 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*, Cristian Danescu-Niculescu-Mizil, Jacob Eisenstein, Kathleen McKeown, and Noah A. Smith (Eds.). Association for Computational Linguistics, Baltimore, MD, USA, 44–48. https://doi.org/10.3115/v1/W14-2513

[41] Saríah Lopez-Fierro and Ha Nguyen. 2024. Making Human-AI Contributions Transparent in Qualitative Coding. (2024). https://repository.isls.org//handle/1/10537 Publisher: International Society of the Learning Sciences.

[42] Andrew Lowe, Anthony C. Norris, A. Jane Farris, and Duncan R. Babbage. 2018. Quantifying Thematic Saturation in Qualitative Data Analysis. *Field Methods* 30, 3 (Aug. 2018), 191–207. https://doi.org/10.1177/1525822X17749386

[43] Penny Mackieson, Aron Shlonsky, and Marie Connolly. 2019. Increasing rigor and reducing bias in qualitative research: A document analysis of parliamentary debates using applied thematic analysis. *Qualitative Social Work* 18, 6 (Nov. 2019), 965–980. https://doi.org/10.1177/1473325018786996

[44] 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 (Nov. 2019), 1–23. https://doi.org/10.1145/3359174

[45] Janet McGaw and Alasdair Vance. 2023. Dissonance, Disagreement, Difference: Challenging Thematic Consensus to Decolonise Grounded Theory. *International Journal of Qualitative Methods* 22 (Jan. 2023), 16094069231220775. https://doi.org/10.1177/16094069231220775

[46] Jane Mills, Ann Bonner, and Karen Francis. 2006. The development of constructivist grounded theory. *International journal of qualitative methods* 5, 1 (2006), 25–35.

[47] Alireza Moghaddam. 2006. Coding issues in grounded theory. *Issues In Educational Research* 16 (2006). https://www.iier.org.au/iier16/moghaddam.html

[48] Michael Muller, Shion Guha, Eric P.S. Baumer, David Mimno, and N. Sadat Shami. 2016. Machine Learning and Grounded Theory Method: Convergence, Divergence, and Combination. In *Proceedings of the 2016 ACM International Conference on Supporting Group Work (GROUP '16)*. Association for Computing Machinery, New York, NY, USA, 3–8. https://doi.org/10.1145/2957276.2957280

[49] Lorelli S. Nowell, Jill M. Norris, Deborah E. White, and Nancy J. Moules. 2017. Thematic Analysis: Striving to Meet the Trustworthiness Criteria. *International Journal of Qualitative Methods* 16, 1 (Dec. 2017), 160940691773384. https://doi.org/10.1177/1609406917733847

[50] Steven Pace. 2004. A grounded theory of the flow experiences of web users. *International Journal of Human-Computer Studies* 60, 3 (March 2004), 317–363. https://doi.org/10.1016/j.ijhcs.2003.08.005

[51] Angelina Parfenova et al. 2024. Automating Qualitative Data Analysis with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*. 177–185.

[52] 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 (Nov. 2016), 102. https://doi.org/10.5539/jel.v6n1p102

[53] Maryam N. Razavi and Lee Iverson. 2006. A grounded theory of information sharing behavior in a personal learning space. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. ACM, Banff Alberta Canada, 459–468. https://doi.org/10.1145/1180875.1180946

[54] David Ribes. 2019. How I Learned What a Domain Was. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (Nov. 2019), 1–12. https://doi.org/10.1145/3359140

[55] 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)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3411764.3445591

[56] John Hamon Salisbury and Tom Cole. [n. d.]. Grounded Theory in Games Research: Making the Case and Exploring the Options. ([n. d.]).

[57] Sina Mahdipour Saravani, Sadaf Ghaffari, Yanye Luther, James Folkestad, and Marcia Moraes. 2023. Automated Code Extraction from Discussion Board Text Dataset. In *Advances in Quantitative Ethnography*, Crina Damşa and Amanda Barany (Eds.). Vol. 1785. Springer Nature Switzerland, Cham, 227–238. https://doi.org/10.1007/978-3-031-31726-2_16 Series Title: Communications in Computer and Information Science.

[58] 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 (July 2018), 1893–1907. https://doi.org/10.1007/s11135-017-0574-8

[59] Nikhil Sharma, Q Vera Liao, and Ziang Xiao. 2024. Generative Echo Chamber? Effect of LLM-Powered Search Systems on Diverse Information Seeking. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–17.

[60] 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*. 63–70.

[61] 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*. ACM, Delft Netherlands, 17–25. https://doi.org/10.1145/3663433.3663456

[62] 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.* (1998), xiii, 312–xiii, 312. Place: Thousand Oaks, CA, US Publisher: Sage Publications, Inc.

[63] Gareth Terry, Nikki Hayfield, Victoria Clarke, and Virginia Braun. 2017. Thematic analysis. *The SAGE handbook of qualitative research in psychology* 2, 17-37 (2017), 25. https://books.google.com/books?hl=en&lr=&id=AAniDgAAQBAJ&oi=fnd&pg=PA17&dq=Thematic+analysis+terry+&ots=dpi2nmHiMV&sig=959tII4BUp9su6Hv2JJui1KjP5Q Publisher: SAGE Publications Ltd.

[64] Nguyen Cao Thanh and TT Thanh. 2015. The interconnection between interpretivist paradigm and qualitative methods in education. *American journal of educational science* 1, 2 (2015), 24–27.

[65] David R. Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation* 27, 2 (June 2006), 237–246. https://doi.org/10.1177/1098214005283748

[66] Anthony G. Tuckett. 2005. Applying thematic analysis theory to practice: A researcher's experience. *Contemporary Nurse* 19, 1-2 (Aug. 2005), 75–87. https://doi.org/10.5172/conu.19.1-2.75

[67] Thomas Übellacker. 2024. AcademiaOS: Automating Grounded Theory Development in Qualitative Research with Large Language Models. *arXiv preprint arXiv:2403.08844* (2024).

[68] Hanna M. Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*. ACM Press, Pittsburgh, Pennsylvania, 977–984. https://doi.org/10.1145/1143844.1143967

[69] Yixin Wan, George Pu, Jiao Sun, Aparna Garimella, Kai-Wei Chang, and Nanyun Peng. 2023. " kelly is a warm person, joseph is a role model": Gender biases in llm-generated reference letters. *arXiv preprint arXiv:2310.09219* (2023).

[70] Carla Willig and Wendy Stainton Rogers. 2017. *The SAGE handbook of qualitative research in psychology*. Sage. https://books.google.com/books?hl=en&lr=&id=AAniDgAAQBAJ&oi=fnd&pg=PR7&dq=The+SAGE+Handbook+of+Qualitative+Research+in+Psychology&ots=dpi2nmHiK_&sig=uyFhaXXsypuO5HjEBxbBLAaNG50

[71] 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)*. Association for Computing Machinery, New York, NY, USA, 75–78. https://doi.org/10.1145/3581754.3584136

[72] 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)*, Golnaz Arastoopour Irgens and Simon Knight (Eds.). Springer Nature Switzerland, Cham, 470–485. https://doi.org/10.1007/978-3-031-47014-1_32

[73] 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)*. IEEE, 164–169.