PaTAT: Human-AI Collaborative Qualitative Coding with Explainable Interactive Rule Synthesis

Simret Araya Gebreegziabher*
University of Notre Dame
Notre Dame, IN, USA
sgebreeg@nd.edu

Zheng Zhang*
University of Notre Dame
Notre Dame, IN, USA
zzhang37@nd.edu

Xiaohang Tang
University of Liverpool
Liverpool, UK
sgxtang4@liverpool.ac.uk

Yihao Meng
Xi'an Jiaotong University
Xi'an, China
ymeng2@nd.edu

Elena Glassman
Harvard University
Cambridge, MA, USA
glassman@seas.harvard.edu

Toby Jia-Jun Li
University of Notre Dame
Notre Dame, IN, USA
toby.j.li@nd.edu

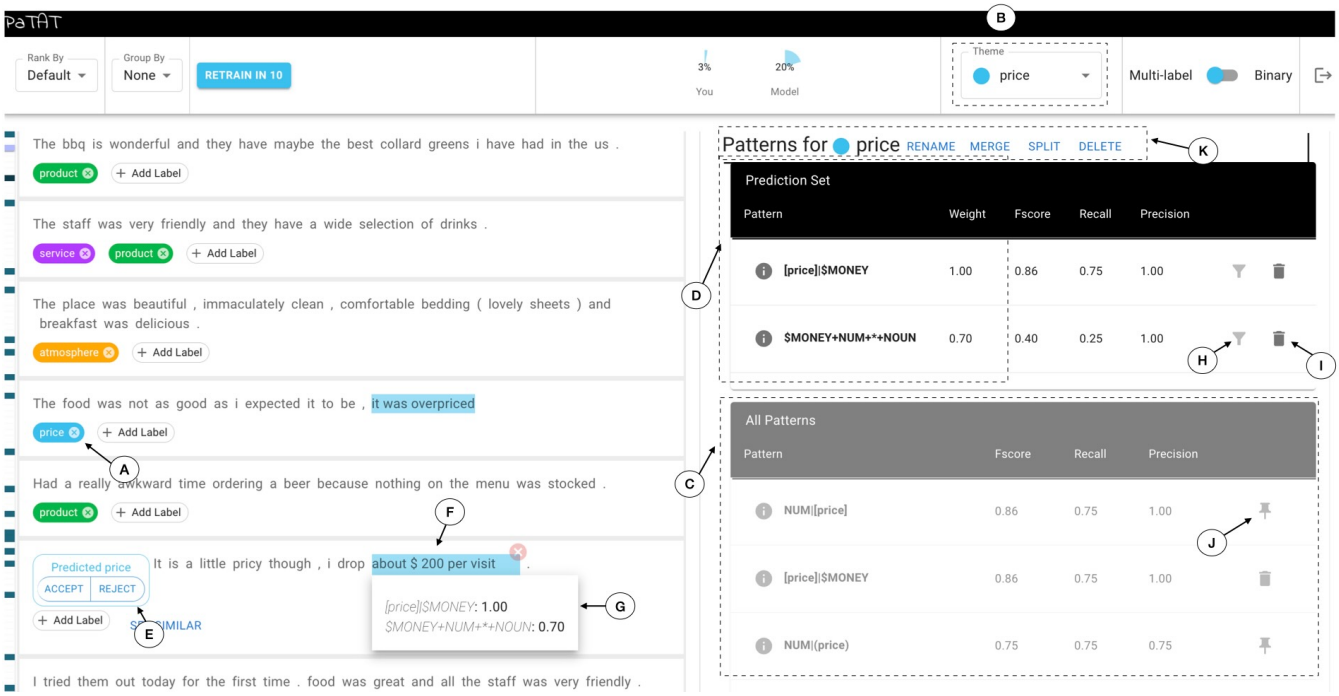

Figure 1: Users start using PaTAT by creating and adding thematic labels to data items (A). For the user-selected theme (B), PaTAT then identifies multiple patterns (C) that best align with the user-assigned labels. To support interpretability, PaTAT selects five or fewer patterns that, when combined in a linear combination (D), most agree with existing user-provided labels. This linear combination is then used to predict labels for data items (E). Phrases that match the patterns in the prediction set are highlighted (F) for user understanding: users can hover over the highlights to see the matched patterns (G). Users have the ability to filter data based on patterns (H), remove spurious patterns (I), and manually promote relevant patterns to be included in the prediction set (J). To support codebook evolution, users can rename, merge, split, or delete themes (K).

ABSTRACT

Over the years, the task of AI-assisted data annotation has seen remarkable advancements. However, a specific type of annotation task, the qualitative coding performed during thematic analysis, has characteristics that make effective human-AI collaboration difficult.

*These authors contributed equally to this work.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581352

Informed by a formative study, we designed PaTAT, a new AI-enabled tool that uses an interactive program synthesis approach to learn flexible and expressive patterns over user-annotated codes in real-time as users annotate data. To accommodate the ambiguous, uncertain, and iterative nature of thematic analysis, the use of user-interpretable patterns allows users to understand and validate what the system has learned, make direct fixes, and easily revise, split, or merge previously annotated codes. This new approach also helps human users to learn data characteristics and form new theories in addition to facilitating the "learning" of the AI model. PaTAT's usefulness and effectiveness were evaluated in a lab user study.

CCS CONCEPTS

• **Human-centered computing → User interface programming**.

KEYWORDS

data annotation, human-AI collaboration, qualitative analysis

ACM Reference Format:

Simret Araya Gebreegziabher, Zheng Zhang, Xiaohang Tang, Yihao Meng, Elena 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), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3544548.3581352

1 INTRODUCTION

Thematic analysis is one of the most common qualitative data analysis methods. Its practitioners search through a dataset to identify, analyze, and report repeated patterns [5]. This method is frequently used not only in human-computer interaction (HCI) but also in a wide range of behavioral and social science research domains. The primary process of thematic analysis is *qualitative coding*, where coders identify the items of interest in the data and assign them codes (labels) [4]. However, coding is usually a laborious and time-consuming task [7, 37, 57] that requires the researcher to go through every item in the data, sometimes multiple times, to annotate data with codes and eventually extract themes of interest. The amount of effort involved makes thematic analysis scale poorly to large corpora [7, 27, 52].

Prior work has explored the use of machine learning (ML) to augment the intelligence of the user during different types of data annotation tasks, including qualitative coding. Previous need-finding studies [16, 27, 40] found great potential in assisting users with qualitative coding by utilizing ML to partially automate some aspects of the user's task. Several recent qualitative data analysis (QDA) systems (e.g., CODY [52], WebAnno [65], and NVivo) use the human-in-the-loop ML approach, where an ML model is trained in real time to predict additional items' labels based on the user's on-going manual labelling. Despite their potential, the limitations of the tools in terms of their flexibility and the level of support they provide for qualitative coding still persist.

Why not use more general ML-powered text annotation tools? First, many existing ML-enabled annotation tools are designed for unambiguous and deductive task domains, e.g., object classification in images and entity extraction in texts, where the set of possible labels remains mostly static and the "correct" label for each item is clear to human annotators. In contrast, the labels developed through qualitative coding usually *emerge* and *evolve* over time as the annotator goes through the data [7, 27] in an inductive way. Codes are frequently revised, merged, or split during this process. There is also hardly the notion of "ground truth" in qualitative coding, as the ultimate goal is *not* to build a machine-learnable model of the data, but to discover themes of interest that help answer specific research questions [7]. Therefore, any AI tool must honor serendipity, human agency, and ambiguity, as argued in Jiang et al. [27], to *assist* users with their current workflow instead of breaking their workflow with automation [16].

Second, the design goal of most human-AI collaborative annotation tools (e.g., [29, 55, 66]) is to optimize the accuracy of the prediction, which most performance evaluations of these tools also focus on. However, prediction accuracy is far from the only important design goal in qualitative coding. Unlike other annotation domains, such as annotating datasets to train supervised machine learning models, applying codes to data is not the ultimate goal in thematic analysis. Instead, it is a way to facilitate the human researcher's process of extracting findings and insights to answer research questions [7, 40]. Therefore, again, when we design human-AI collaborative approaches to support qualitative coding, we should specifically focus on optimizing for human learning, i.e., providing support for human researchers to (1) understand key patterns in data and (2) identify potential phenomena of interest during the annotation process.

We argue that the existing limitations of ML-based annotation assistants are largely due to the "black box" nature of most ML models, such as end-to-end neural networks (used in [44, 67]) and support vector machines (SVMs, used in [21]). These models lack (1) the support for explanations that help users comprehend the model's recommendations and (2) the transparency that allows users to understand what the model has learned. Although these "black box" models may perform well in terms of prediction accuracy, it is not the only important measure in qualitative coding, as we have discussed. Better explanations and transparency would enable users to understand why a recommendation is made, validate model results, verify that the model uses reasonable and relevant data features, and provide useful affordances to revise and improve model behaviors in representations that the system can understand. All these improvements might serve the aforementioned goals of (1) allowing users to iteratively explore patterns and themes, (2) improving human learning, and (3) enabling more effective and efficient qualitative coding for thematic analysis. Therefore, sacrificing some model accuracy for better explainability could be a worthwhile trade-off in such a human-AI collaborative scenario.

A promising way to achieve better explainability in AI assistance for qualitative coding is to learn *pattern rules* to predict labels for each item. Previous pattern-based systems such as Cody [52] have been developed based on the findings of previous empirical studies [40, 51]. These studies identified AI's interactive learning of rule-based patterns of different labels as a useful way to assist human researchers in the qualitative coding process. Cody's implementation of this approach was shown to be helpful in improving users' understanding of data, increasing coding quality, and accelerating the

coding process in its user evaluation [52]. However, tools like Cody lack flexibility in their syntax of rules to achieve the adequate expressiveness needed for many qualitative coding tasks. For example, Cody's syntax only supports the use of literal or lemmatized words, Boolean operators (AND and OR), and wildcard (∗) characters. This syntax is not sufficient to capture rich lexical (e.g., part of speech tags), syntactic (e.g., structure), and semantic (e.g., word embeddings) information in text, which has been found to be useful in qualitative coding in previous experiments. Due to these limitations of its rule-based approach, Cody used a black-box stochastic gradient descent (SGD) model alongside rules to suggest labels, which further limits the explainability and user control, as the recommendations from SGD cannot be directly interpreted or manipulated by the user.

To address these gaps, this paper presents a human-AI collaborative qualitative coding system named PaTAT.[1] PaTAT[2] introduces an interactive program synthesis approach that learns symbolic sets of pattern rules over neurally-generated lexical, syntactic, and semantic features of the data through a collaborative annotation process with users. These features include parts-of-speech tags, word lemmas, soft matches, entity types, and wildcards. By creating a linear combination of a small set of these pattern rules, PaTAT exploits the power and flexibility of neural methods while maintaining explicit, concise, top-level rule-based reasoning. Furthermore, the interactive program synthesis approach used in PaTAT is designed to specifically accommodate the iterative and ambiguous nature of qualitative coding in thematic analysis, where codes usually emerge and evolve over time and are frequently revised, merged, or split. This design of PaTAT is informed by the insights from previous inquiries of user needs and preferences for AI-enabled assistance in qualitative coding [2, 27], as well as the findings of our formative study with qualitative researchers.

We illustrate the usability of PaTAT through a lab user study with eight qualitative researchers. Participants found PaTAT usable and useful for their qualitative analysis workflow. The findings of this study also suggest design implications for future AI-enabled qualitative analysis-assisting tools, e.g., regarding adapting different models and interaction strategies based on task goals and contexts while working with ambiguities and uncertainties.

This paper makes the following contributions:

(1) A new interactive program synthesis approach that learns symbolic representation of codes and themes through patterns that capture the lexical, syntactic, and semantic features from the user's qualitative coding process in real-time.
(2) PaTAT, a practical system that implements this approach with a mixed-initiative interface for effective human-AI collaboration in thematic analysis that facilitates not only the performance of the annotation but also the learning of its users.
(3) A lab user study with 8 experienced qualitative researchers that demonstrate the usefulness, usability, and effectiveness of PaTAT.

[1]PaTAT is the Dutch word for French Fries and also an acronym for **Pa**ttern-based **T**hematic **A**nnotation **T**ool.
[2]https://github.com/SimretA/PaTAT-pattern-based-thematic-annotation-tool

2 RELATED WORK

2.1 Qualitative Coding in Thematic Analysis

Qualitative coding is one of the most important aspects of thematic analysis [56]. It is the process of indexing, categorizing, and labeling qualitative data [49]. The goal of the process is to interpret the data in relation to some research question(s) [22]. In qualitative coding, annotators tag portions of the data that are relevant to a particular theme. To do this, the researcher searches the dataset to identify shared traits and patterns of meaning and assigns themes to them [4, 60]. This requires researchers to make multiple passes of the data to identify not only individual themes, but also relationships and links between them [20]. Different forms of collaboration can take place during these passes. While researchers start collaboration over a small sample of the dataset to come to a consensus on the codebook, later stages can be accompanied by collaboration to maintain consistency [19].

Different levels of analysis require different coding approaches. Researchers often start by providing low-inference codes closely related to the data [38]. In further analysis, themes can be iteratively developed through the analysis of codes rather than just data [49], resulting in more interpretive higher-order themes. For this reason, themes are always formed, dissolved, and revised during qualitative coding [61]. In a three-phase qualitative analysis of text data, Winters et al. [61] found that new codes were generated as researchers gained new insights from their data. The complexity of the codebook changed throughout the three phases as new sub-themes emerge and grow out of defined themes. PaTAT's interaction model allows users to refine existing themes using two methods: (1) using examples and (2) using generated rules. This allows researchers to make appropriate theme refinements for the various levels of analysis.

Qualitative coding can be especially challenging and time consuming because it requires dynamic and creative inputs from researchers [2]. Furthermore, because the foundational analysis is not theoretically bounded [5], qualitative coding must have a flexible account of the data. The role of qualitative coding in thematic analysis is not merely for the result of labeled data, but importantly, for the learning of researchers to discover, support, and validate their theories from the data [2]. By offering a variety of grouping and ranking strategies and displaying what the system has learned about the data through interpretable patterns, PaTAT enables users to obtain a more comprehensive overview of the data. Additionally, it supports the visualization of the data's theme distribution, allowing users to understand high-level data trends.

Nelson [46] presents a three-step approach (pattern detection, refinement, and confirmation) for scalable and inductive exploratory analysis that involves subjective input from a human user, as well as the computational power of computers, taking into account the evolution of information discovered by researchers. In light of this, PaTAT provides users with full agency and control of the qualitative coding process through the ability to revise model-generated rules and user-defined codes, as well as the flexibility to do it at varying granularity levels. Through its suggestions of which codes might apply to which data items, PaTAT offers a collaborative coding interaction between the user and the AI by learning the user's annotating patterns.

2.2 Tools for Qualitative Data Analysis

Researchers working with qualitative data analysis (QDA) use tools to examine data collected using various methods, such as interviews, focus groups, and field notes. Although the use of QDA tools is common in fields such as health sciences, humanities, and social sciences, the vast majority of these tools are used for data management rather than to assist researchers with their methods and findings [62]. According to Woods et al. [62], researchers who used QDA tools during the analysis phase tended to emphasize codebook management and ease of retrieval instead of knowledge discovery. Popular QDA tools like NVivo, MAXQDA, and ATLAS.ti allow search by keywords and batch annotation of data points to accelerate the coding process. These tools have also been used to create links between data sections and write memos [70]. Although these tools provide valuable assistance with basic code organizing tasks, they are not very effective in providing assistance to the user's learning.

Several new ML-assisted QDA tools have been developed to help researchers find themes, develop codebooks, and automatically label data points in qualitative analysis. INCEpTION [29] and BRAT [55] offer semi-automated assistive coding, with the aim of improving the efficiency and quality of coding by incorporating active learning. These tools rely on the user to iteratively train a predictive model that will recommend new labels to unlabeled data points. Similarly, Li et al. [36] takes an active learning approach by utilizing a pre-trained language model to save the labor of annotation while leveraging human input to repair the machine's mistakes. Although these approaches show promising results in terms of the efficiency and accuracy of the model, they fall short in providing users with the information and flexibility required for iterative theme discovery in qualitative coding. Furthermore, their limited interaction design and black-box nature makes it difficult for users to contextualize, refine, and rediscover themes [3]. Guetterman et al. [23] found that Natural Language Processing (NLP) methods are useful in identifying the main themes in traditional qualitative analysis. Although this technique was shown to accelerate the analysis process for researchers, it cannot uncover the crucial context and subtleties in the data. This can lead to overlooking important details in the data.

Rule-based text analysis is a promising approach for analyzing unstructured textual data [12]. The rule-based approach can provide the easy interpretation and mutability that qualitative researchers are looking for [27]. For example, Tempura [63] uses context-preserving structural templates generated from the dataset to group and analyze search queries. It takes advantage of the repetitive and condensed nature of search queries to linguistically summarize them. Although Tempura provides a linguistically extensive rule set that includes named entities and part-of-speech tags with an interface for interactive data exploration, its templates lack flexibility because they are targeted at well-structured data.

Another rule-based qualitative coding approach combines rule-based approaches with ML techniques to enhance the coding process. Cody [52], a web-based qualitative coding tool, uses rule-based and ML techniques to help researchers identify patterns and codes during qualitative analysis. Cody uses a pre-trained language model to generate initial rules that include part-of-speech tags and word lemmatization for user-defined codes and data annotations. An ML model is then trained using the generated rules to label unseen data points. Cody's increased user involvement and ability to structure and highlight data supports user insight and speeds up the coding process. However, we believe that the lack of expressiveness possible within its rules limits their versatility. For example, Cody's syntax only supports the use of literal or lemmatized words, Boolean operators (AND and OR), and wildcard (∗) characters. This syntax is not sufficient to capture rich lexical (e.g., part of speech tags), syntactic (e.g., structure), and semantic (e.g., word embeddings) information in text, which has been found useful in qualitative coding in previous experiments [10, 11]. Therefore, PaTAT provides a more extensive pattern language. Compared to Tempura or Cody, PaTATalso presents new interface features for users to explore the model's state of learning; understand the reasons behind code recommendations for individual data items; evolve the codebook by merging, splitting, or revising codes; discover new themes; and understand the high-level trends and characteristics of the dataset.

2.3 ML-Assisted Data Annotation Tools

With the growing need to annotate large datasets for supervised ML applications, many ML-assisted data annotation tools have been employed to speed up manual data annotation [9, 34, 59, 66]. Human-in-the-loop semi-automated annotation tools usually use an active learning approach to recommend labels for unlabeled items and support batch labeling [69], which can improve the quality of annotations while reducing the costs and efforts required compared to manual annotation [30]. These annotation tools are optimized for the efficiency of classification and predictive tasks [34].

Although the goals of qualitative coding and supervised classification problems may initially appear similar, they have quite different objectives. Qualitative coding differs from classification tasks due to the extra complexity of dynamically changing labels and the need to allow users to systematically draw inferences. The hypothesis-driven nature of qualitative coding can benefit a lot from the researchers' prior knowledge [53] while existing ML-assisted data annotation tools [58, 64] often fail to support the exploratory, ambiguous, and subjective nature of qualitative analysis. Due to this misalignment in their goals, Chen et al. [6] discovered that data labels annotated for training strong ML classifiers may not be useful for qualitative researchers.

2.4 Interactive Program Synthesis

To learn interpretable rules that describe themes in annotation, PaTAT uses *program synthesis*, the technique of automatically generating a sequence of computer executable rules from a high-level specification [25]. A common way of providing such specifications is programming by example (PBE), where users provide desired behaviors of a system in terms of inputs and outputs. In PaTAT, the goal of PBE is to learn rule patterns that describe a theme from examples of positive items that belong to this theme and negative items that do not belong to this theme.

Inductive program synthesis automatically finds programs that satisfy the user's intent, as stated in positive and negative input-output examples [13, 25]. PBE can be an interactive process in which

new examples are added after each iteration to help clarify the requirements [24, 25]. The cognitive burden of users can be reduced while producing efficient programs by providing users with the ability to continuously and iteratively clarify their intentions [32]. The incomplete specification requirement and continuous refinement in program synthesis can also provide flexible user interaction [17], making it ideal for repetitive and interactive tasks such as qualitative coding.

A primary issue with the use of PBE for text analysis is the large number of examples needed to synthesize generalizable programs [8]. The design of PaTAT's interaction model is informed by several key design strategies in Zhang et al.'s [68] work. Zhang et al. [68] propose an interaction model that addresses the overfitting problem by allowing users to specify which parts of the provided examples should be generalized. In relation to this, PaTAT's phrase-level annotation allows users to specify which parts of a sentence make it a positive (or negative) match, allowing the synthesizer to learn distinguishing features at a finer granularity. Another major challenge in program synthesis is to efficiently search for the desired program. To address the performance challenge, some previous efforts seek to reduce the initial search space [1, 54] while others use statistical approaches to condense the potential pool of programs [33]. By utilizing information gain heuristics over its versatile specification language, PaTAT is able to synthesize abstract and expressive programs while effectively scoping the search space.

3 FORMATIVE STUDY

To better understand the current practices and challenges of users and their needs and preferences for AI assistance in qualitative coding, we conducted a formative contextual inquiry study.[3] We observed the current qualitative coding workflow of people with thematic analysis experience, and interviewed them about it. In particular, the formative study gathered information on (1) qualitative coders' daily practices of analyzing and coding qualitative data in thematic analysis, (2) their experiences and opinions of existing QDA assistance tools, and (3) their needs and goals for AI assistance in QDA.

3.1 Participants

We recruited five participants (2 women, 2 men, and 1 non-binary participant) for this study. Each participant had at least two years of experience in qualitative coding and thematic analysis. All participants were researchers who frequently use thematic analysis in their projects.

Our participants reported using different tools for their qualitative coding tasks (e.g., a spreadsheet, Figma[4], Mural[5], Miro[6], or Saturate[7]). One of the sessions was held as three participants collaborated to analyze interview data for the first time, while others consisted of participants going over their previously coded data again.

[3]The study protocol has been reviewed and approved by the IRB at the lead author's institution, where the study was conducted.
[4]https://www.figma.com/
[5]https://www.mural.co/
[6]https://miro.com/
[7]http://www.saturateapp.com/

3.2 Study Protocol

We used the contextual inquiry method [28] to gain insight by observing the usual coding practices of the participants and asking them key questions during coding sessions and follow-up interviews. Each session began with the informed consent process, followed by an introduction to the objectives of the formative study. Then, we asked each participant to perform qualitative coding on representative (anonymized) data from their own research as they would normally. The coding session lasted 30 minutes to an hour. As the participant performed qualitative coding, we asked questions about their codebooks, the strategies they were using to come up with their codes, and the ways they used their tools of choice.

Following the coding session, we interviewed the participant to learn more about their analysis goals, the methods they used, and their coding practices. We asked the participants to elaborate on the decisions they made during the task. We discussed their answers to questions about why they chose the tools they were using, how they handled collaborations and contradictions, what strategies they used to increase their efficiency and reduce fatigue, and what they would expect from AI assistance for qualitative coding. The interviews lasted about an hour for each participant and were carried out virtually over Google Meet or Zoom.

3.3 Analysis Methods

The data collected from these interviews was explored using thematic analysis. First, two members of the research team independently examined the raw data to derive lower-level codes. After that, the team met to cross-check and validate the codes and compiled a unified codebook that was used to analyze the data. Although certain themes about ease of use and control emerged consistently throughout the interviews, some themes were more specific to the goals the researchers had in mind.

3.4 Key Insights

Our formative study found that participants often used a hypothesis-driven approach to initiate the coding process, while maintaining the flexibility to iteratively evolve the codes. Although participants valued the assistance offered by intelligent QDA tools, they did not want to sacrifice control and leadership over the coding process. Instead, participants sought effective means to help them inspect and understand the rationale behind automated coding so that they could decide what to *trust* and what they could *learn* from their AI counterparts. Lastly, participants expected QDA tools to play a collaborator role that is able to improve and expand coders' capabilities in the human-AI collaboration.

3.4.1 ***KI1****: Participants valued finding new insights about their data from qualitative coding.* Participants often began their coding process with some anticipated findings. From our interviews, we understood that although participants entered the coding process with some hypotheses in mind, they also used this phase of the analysis to discover emerging themes and to gain a broad intuition about their data and projects. The necessity for a greater degree of abstraction to support finding additional ideas at each iteration of the coding process was consistently emphasized in our interactions with all participants. For example, a participant viewed the

different passes of the coding process as follows: *"I always start with the surface hints for annotation, but it's more rigorous really I want to uncover more than a surface level understanding of the data. I also really want to try to see how related themes can connect. Or maybe not really related at a surface level but like deeper relationships"* (P5). Another participant also emphasized that themes commonly emerge and keep changing from the initial idea they had: *"For most of our projects we had some initial idea about our results, but after we organized our cookbook we always found that, Oh, this is also a really great idea, maybe this can be part of our next project or maybe we can also mention it in our current project. For every step, we define so many new themes, and you find new ideas about decoding"* (P4).

3.4.2 ***KI2****: Participants valued agency and autonomy in the coding process.* Participants expected an assistive tool to recommend potential themes to use for data items as it begins to learn. Although most of the participants were concerned that context and details would be lost in abstraction, they valued having a tool that would relieve them of repetitive activities in the coding process: *"I still would like to make the codebook first and then give it to the system and it will do its thing and then give me suggestions. You know, I want to contrast what I found with what the system thinks. [The system] should not do coding instead of me, but maybe in addition to me"* (P5).

3.4.3 ***KI3****: Participants valued transparency and trust in assistive QDA tools.* Participants emphasized that understanding the process and being able to see how and what the tool is learning is an important factor to establish their trust in the tool. The need for parallelism between annotation and forming conclusions and theories, as well as a clear means for the tool to demonstrate how certain inputs were transformed into outputs, was brought up by a few participants. The participants stressed the importance of being able to correct the tool and validate that their corrections propagate throughout the entire dataset: *" I would want it to be able to learn, and I want to see how it learned if that makes sense. Train it again and kind of like give my feedback to it, detailed enough that made me feel confident it is going to learn if it takes them"* (P1). At the same time, participants want the explanation medium to be easy-to-understand: "*the way of presenting explanation[s] should be straightforward for people like me who have little knowledge about AI*" (P2).

3.4.4 ***KI4****: Participants expected the tool to act as a collaborator or an extension to their abilities.* Participants made it clear that although they wanted to be the primary annotator in the process, they wanted the tool to help improve and expand their capabilities: *"I think it will almost be like another team member, so you know I would want to be able to have collaborative features like with a real person I made the cookbook with... My collaborators usually don't have the time to like you know go through line by line and everything, so it's up to me to make sure I'm thorough. So having this tool that would have another layer of like your angle to look at things and also just another layer of assurance that I am really going through everything"* (P5).

3.5 Design Goals

We summarize the following five key design goals from the findings of our formative study, which helped guide the design of PaTAT described in Section 4:

- **DG1**: Improve user's understanding of the dataset, i.e., help them (1) discover low-level nuance and commonalities across data items and (2) develop high-level mental models of the dataset (Section 3.4.1).
- **DG2**: Generate explainable coding recommendations that allow users to understand the AI's rationale, verify the soundness of the AI's reasoning, and provide affordances for the user to correct the AI's current model (Section 3.4.3).
- **DG3**: Accommodate the iterative and ambiguous nature of qualitative coding at different granularity levels (Section 3.4.1)
- **DG4**: Ensure user agency in the coding process so that users have the authority to make the ultimate annotation decisions when necessary (Section 3.4.2)
- **DG5**: Facilitate effective human-AI collaboration to resolve uncertainty and ambiguity in annotations (Section 3.4.4)

4 PATAT SYSTEM

To address the five design goals, we designed and implemented PaTAT, an AI-powered qualitative data annotation tool that learns explainable label recommendations, supports the iterative discovery and revision of themes, and facilitates users' understanding of the data. PaTAT provides several features that allow flexible human-AI collaborative coding in which both parties can improve their understanding of data through mixed-initiative interaction, while users still have full control of the entire coding process.

The main interfaces of PaTAT are shown in Fig. 3. The interface consists of three main UI components: (1) A toolbar that allows for a variety of configurations (Fig. 3-A); (2) A coding panel where users can perform qualitative coding in collaboration with AI (Fig. 3-B); (3) A pattern rule panel that displays synthesized rules and supports rule inspection and manipulation (Fig. 3-C).

The architecture and the human-AI collaborative workflow of PaTAT are illustrated in Figure 2. As shown, the user interacts with an interactive rule synthesizer through PaTAT's interface by iteratively annotating, grouping, and/or ranking data; reviewing explanations of model recommendations and synthesized rules; and repairing model outcomes when needed.

In this section, we will start by describing an example user scenario. Then we will discuss the key features of PaTAT and the algorithms that implement them.

4.1 Example User Scenario

Suppose that Alan wants to conduct a thematic analysis to understand the different themes in user opinions about local businesses. To do so, he plans to perform qualitative coding on a set of customer reviews of local businesses on Yelp, a popular review site. To start, he loads the reviews into PaTAT. PaTAT will display the list of reviews in the coding panel (Fig. 3-1B) for user annotation. Alan can start working through the list by analyzing each item, coming up with new codes if applicable, and assigning one or more codes to each item. As Alan creates more codes and makes more annotations, a

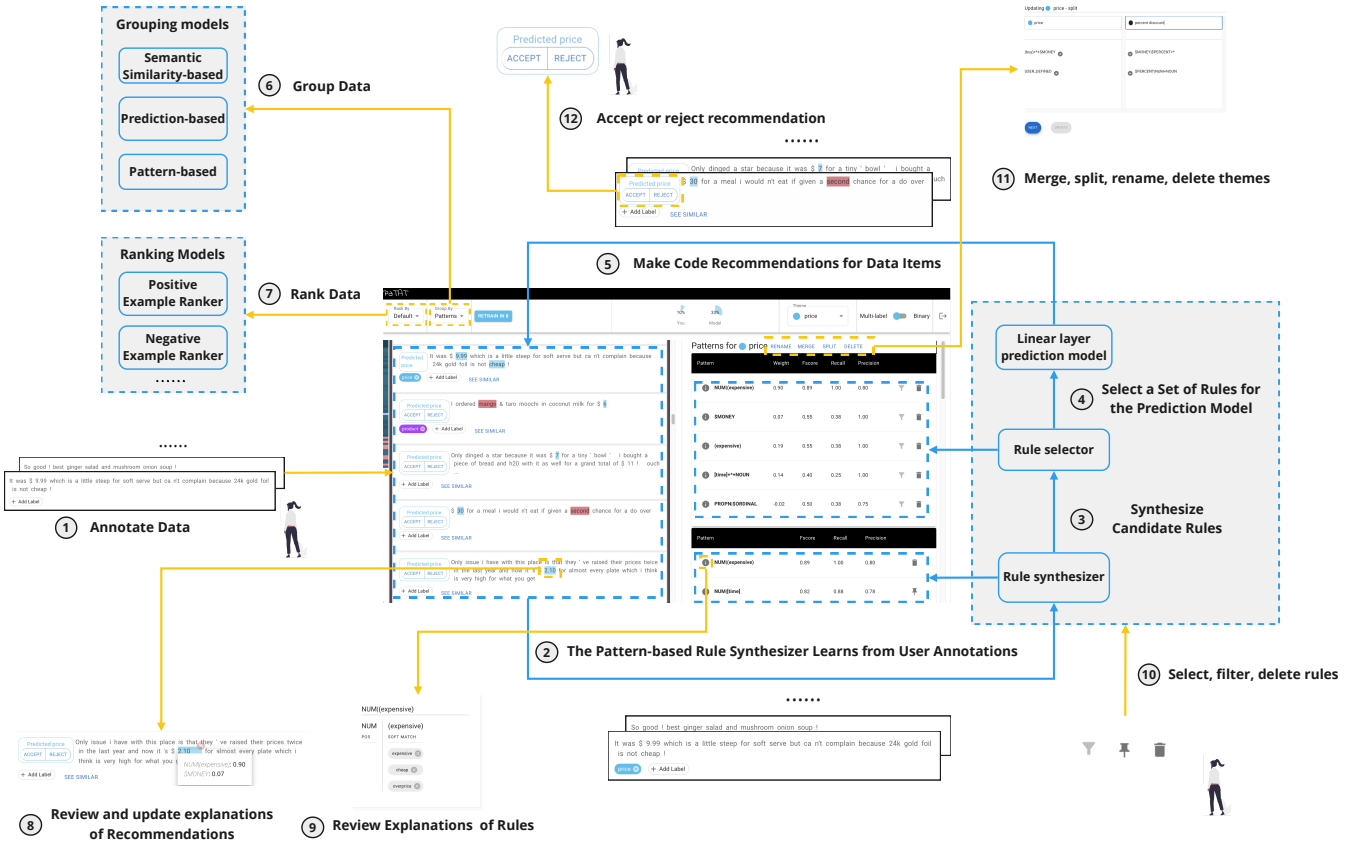

Figure 2: The system architecture of PaTAT: While (1) the user starts annotating data on the left panel, (2) PaTAT learns patterns from the user's annotations by (3) synthesizing rules over parts of speech, literal words, word lemmas, soft matches, entity types, and wild cards. (4) PaTAT selects a small set of rules that can capture the broadest of positive annotations and trains a linear layer model. Then PaTAT (5) recommends codes for data items in the left panel based on that linear model over patterns. The user has the option to group (6) and rank (7) data based on different strategies. The user can give feedback to the model by reviewing and updating recommendations (8), reviewing and updating rules (9), and selecting and removing rules (10). The user can iteratively update the codebook (11), and make annotations by accepting/rejecting recommendations (12) Yellow lines represent users' interaction, while blue lines represent PaTAT's information flow.

program synthesizer (described in Section 4.3.1) learns interpretable patterns that describe these codes and predict items that can be coded with them. The synthesized patterns are displayed on the right panel (Fig. 3-1C). The predicted labels are displayed at the beginning of each item in the coding panel (Fig. 3-1B). For each prediction, Alan can see an explanation through the highlights of matched patterns (Fig. 3-3): themed color highlights indicate patterns that make an item more likely to be a positive case for the current code, while red highlights mean the opposite. Alan can also hover over a recommendation to view a list of all patterns that matched this item. When Alan clicks on a pattern in Fig. 3-1D, he can see a list of all the items that this particular pattern matches. The panel also displays the performance statistics of the current model (precision, recall, and F1 score) to fit the already labeled data as a reference.

A crucial part of any human-AI collaborative task is error discovery and repair, and our scenario is no exception. The goal of PaTAT is to allow the user to *understand* and provide *direct input* to what the model has learned. If Alan does not agree with a code recommended by the system, he can reject it, and the model will retrain to learn from Alan's feedback. Since the synthesized patterns are understandable by users, Alan may also review the list of synthesized patterns to gain a direct understanding of what the system has learned. If he disagrees with a pattern (e.g., the pattern describes a feature that is clearly irrelevant to the current theme), he can reject it using the panel Fig. 3-1E. Similarly, he can "promote" the use of a synthesized pattern that he considers to be highly relevant (Fig. 3-1F). Lastly, a common challenge in interactive machine learning is dealing with predictions that are "correct for the wrong reason" when there is a lack of transparency in model predictions. In PaTAT, Alan can identify such instances by looking for highlighted portions of text that are not relevant to the code. Then he can fix them by selecting the relevant portions instead, allowing

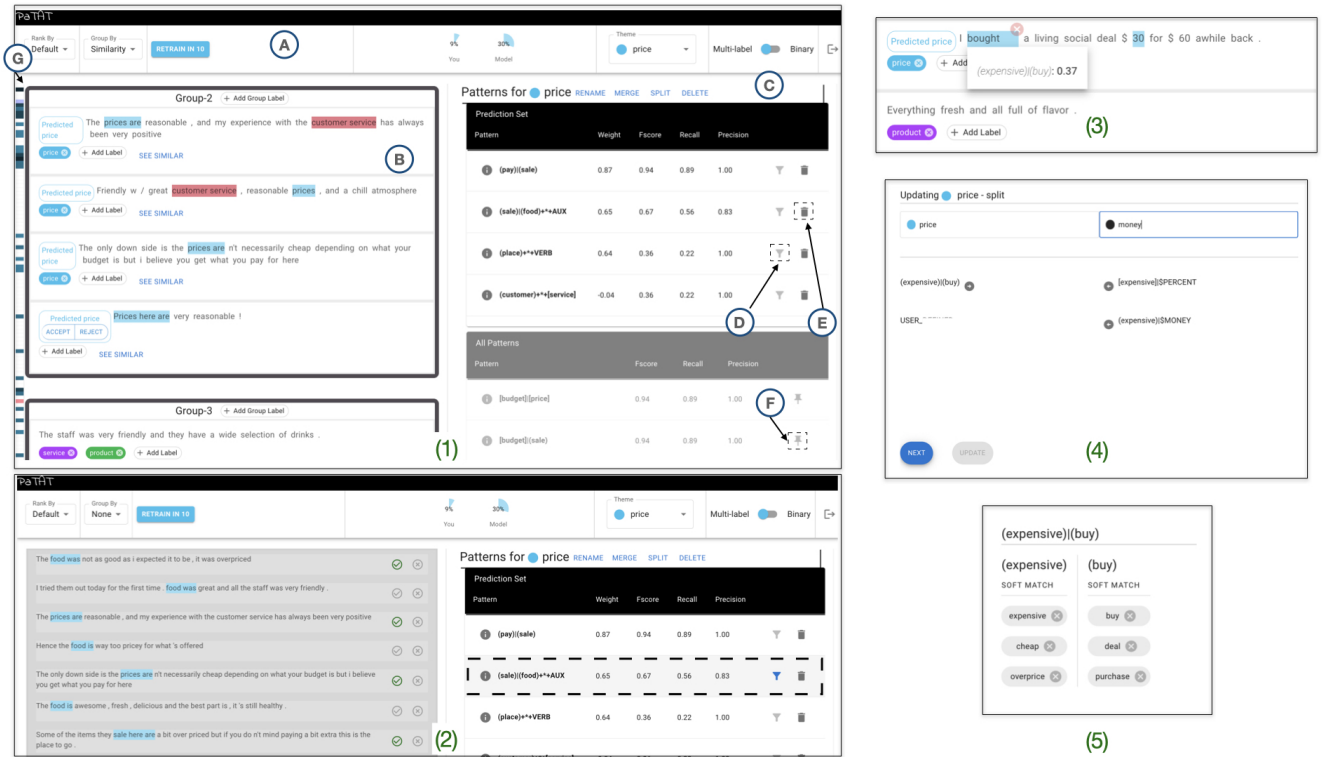

Figure 3: PaTAT provides different methods for organizing and interacting with data: (1) The dataset can be grouped by thematic similarity. (2) The pattern filter shows all the items that matched that pattern. (3) When a user hovers over a highlighted item, all matching patterns are displayed. (4) As the user discovers new sub-themes within a theme, they have the option to split the theme using the patterns associated with the theme. (5) Users can directly manipulate patterns by dictating what synonyms (soft-matches) are allowed to match.

the model to learn not only from the ground truth labels provided by the user but also the reason *why* a user selected a particular label.

As discussed previously, a critical feature of qualitative coding is its ambiguous, subjective, and iterative nature. Unlike most other annotation tasks with fixed sets of labels, codes in qualitative coding often change over time as the user's understanding of the data and the domain evolves. Alan can use the theme adjustment panel (Fig. 3-4) to merge multiple codes into one, split a code into multiple codes, or rename a code as needed, e.g., splitting a code on "service quality" into "friendliness" and "promptness". A benefit of using human-interpretable patterns is that Alan can split a code by patterns instead of relabeling all individual items. Patterns for a theme often naturally represent different semantic clusters within a theme, facilitating the discovery of code-revision opportunities. After the merge, split, or revision of codes, the model will automatically retrain to reflect new changes.

Lastly, while Alan is helping the *model* learn more about the data so that it can better assist him with his qualitative coding task, his most important goal is to understand the characteristics and trends of the data *himself* so that he can gain insights and form theories—the objectives of qualitative coding in thematic analysis. To facilitate *his learning*, Alan can use the grouping mechanism to group similar items (by patterns matched, model-predicted labels, or semantic similarity as described in Section 4.2.2) or use different ranking strategies (e.g., the confidence level of the model as described in Section 4.2.2) to re-rank the list. These strategies enable Alan to better discover trends, clusters, and outliers in the data. Alan can also filter the data using matched patterns or retrieve a list of similar data items for an item of interest. Lastly, the different colors on the scroll bar (Fig. 3-1G) indicate the labels predicted by the model and the confidence level of each prediction, allowing Alan to see a visualized distribution of codes and themes to observe trends, which is especially useful in datasets with meaningful orders.

4.2 Key Features

In this section, we explain the key features of PaTAT that were demonstrated in the example scenario, including interpretable coding recommendations, strategies to improve users' understanding of data, and support for the iterative coding process.

4.2.1 Interpretable code recommendations. As identified in the formative study (Section 3.4.3), although qualitative researchers recognize the usefulness of automated coding recommendations, their acceptance of the recommendations depends on how well they

understand the rationale behind AI recommendations. PaTAT addresses this need by providing explainable pattern-based code recommendations and several accompanying interaction features that help users comprehend, validate, and repair (if needed) the code recommendations provided by AI (DG2).

Learning human-readable patterns for predicting codes from user annotations. To achieve interpretable code recommendations, PaTAT learns explainable pattern rules that capture lexical, syntactic, and semantic patterns across data items (e.g., sentences, short paragraphs) that were annotated with the same code by users (see Section 4.3). The pattern-learning process occurs simultaneously in the background as the user codes more data. PaTAT updates the learned patterns either when the user makes every 10 new annotations or when the user explicitly clicks the "Retrain" button on the toolbar (Fig. 3-1A). This gives the user some control over the training frequency and minimizes interruption of the user's coding process (DG4).

When the retraining process is complete, the updated patterns are shown in the pattern panel (Fig. 3-1C). The lower list in the panel displays all the synthesized candidate patterns. PaTAT will, by default, select the top 5 patterns that, as a set, would achieve the highest F1 score in matching data items to their currently assigned codes; PaTAT uses a linear layer to combine these top 5 patterns to predict the likelihood that each data item belongs to the current code (details in Section 4.3). PaTAT's linear layer approach allows it to describe codes with more complex semantic meanings that could not be captured with a single rule or simple set of rules. As the user adds more positive and negative annotations, the learned patterns evolve from general patterns (e.g., `ADJ` which would match any adjective) to more abstract and selective patterns (e.g., `[discount]|(expensive)+NOUN` which would match either the word "discount" or synonyms of "expensive" followed by a noun). In addition, the set of top patterns that are selected to make up the linear layer change as more example annotations are provided by the user.

Enabling human comprehension and validation of code recommendations. Although pattern-based recommendations provide explainability (as informed by the results of the formative study in Section 3.4.3), the use of patterns to explain automated annotation still presents a high learning curve for users. To address this barrier, PaTAT provides a variety of interaction strategies to help users understand the synthesized patterns and evaluate the AI's model (DG2). First, PaTAT explains the meaning of each pattern to users. As mentioned in Section 4.1, users can view the user-friendly interpretation of each pattern component by hovering over the info (ⓘ) icon besides each pattern. For example, users can see the specific words that a `soft-match` expression matches in the pop-up tip (Fig. 3-5). To help users understand the quantitative performance of each individual pattern, PaTAT shows the F1, recall, and precision score of each. These scores indicate how well the pattern describes the current annotations of the users. Furthermore, users can click on the filter (▼) icon (Fig. 3-1D) button next to a pattern to see the list of data items that satisfy this pattern. For each pattern, PaTAT also shows its weight parameter in the trained linear layer model, which indicates how much it contributes to the AI's recommendation in order to help users understand the relative importance of this pattern. To explain why a code is recommended for an item, PaTAT highlights portions of the text (e.g., words, phrases) that match a pattern currently being used for prediction. Users can hover over highlighted text to see which pattern applies to it (Fig. 3-3).

Supporting user feedback on code recommendations and model learning. With the model's recommendations, PaTAT allows the user to make the final decision in annotations (DG4). In particular, the user can accept or reject the recommendation on each item by clicking the "Accept" or "Reject" buttons. If a recommendation is rejected, PaTAT will treat the corresponding item as a negative example. As a pattern matches more negative examples, it becomes less feasible and gets pruned. In addition, PaTAT enables mixed-initiative interactions between users and the model so that users can fix errors in the model's recommendations and resolve uncertainties (DG5). For pattern selection, the user can review all candidate patterns and click the pin (📌) icon (Fig. 3-1F) to include a candidate pattern in the prediction set or click on the delete (🗑) icon (Fig. 3-1E) to remove a rule from the prediction set. The code recommendations will be updated and refreshed each time the patterns selected for prediction change. In addition, PaTAT also allows users to indicate why an individual item should be annotated (or not) with the current code. The user can highlight a span of text within an item to constrain the AI to considering patterns that match that specific region, reducing the search space in the rule synthesis phase (details in Section 4.3).

4.2.2 Improving users' learning about data. The results of our formative study suggest that qualitative researchers wish to have tools that not only recommend codes but also help them learn more about the characteristics, trends, and patterns of the data to find new insights and develop new theories (Section 3.4.1). PaTAT accommodates this need by providing users with different data grouping, data ranking, and data exploration strategies (DG1).

Data grouping strategies. Qualitative coding analysis requires users to identify commonalities and disparities among a large number of data items. To facilitate this, PaTAT allows the user to organize data items into clusters using three different grouping strategies: semantic similarity-based grouping, model prediction-based grouping, and pattern-based grouping (Section 4.3.2). Supporting multiple grouping strategies allows users to compare data from different angles to increase the chances of discovering new insights about the data. As the user selects a grouping strategy from the toolbar, PaTAT will show each group of data in a separate sub-panel (Fig. 3-1). Users can also assign a code to all items in a group.

Data ranking strategies. Similar to data grouping strategies, PaTAT also offers several different options for ranking (ordering) the data items. The user can view either the items whose predictions (positive or negative) the model is most confident about or those with predictions that the model is least confident about. The "most confident" order allows users to see items that the model considers to clearly fall into the current code (or the opposite) to understand the model's learned characteristics about the code. On the other hand, the "least confident" order allows the user to understand the ambiguous cases at the borderline of codes according to current

model, helping the user find opportunities to further revise or develop the codes. Details of these ranking algorithms are described in Section 4.3.2.

Simultaneously, the colored bar on the left (Fig. 3-1G) shows the distribution of the model's recommended codes and confidence levels, where a deep hue in the theme color of a code indicates that the model is confident about predicting this code for the corresponding data item.

4.2.3 Supporting the iterative evolution of codebooks. Qualitative coding is an iterative process in which users develop the codebook by incrementally abstracting information and iteratively updating the codes and their meanings as they deepen their understanding of the data and discover new themes. PaTAT is designed to facilitate this intrinsic ambiguous, subjective, and iterative process for users by supporting the easy addition, removal, splitting, and merging of codes (DG3).

In our formative study, the participants pointed out that revising and reorganizing codes is inevitable but laborious work. PaTAT allows users to easily add, remove, split, and merge codes, and these changes will automatically propagate to the relevant data without a full manual relabeling. For example, when users want to split an existing code into two, they can click the "split" button to create a new code in the pop-up window, provide a name for the new code, and drag the corresponding patterns from the corresponding area of the old code to that of the new one (Fig. 3-4). PaTAT will retrain the model and re-predict labels for items that were previously labeled with the original code. The user can validate the results and make adjustments as needed.

4.3 Algorithms

To support the key features, we developed a pattern synthesizer that generates—from user annotations—candidate patterns that might explain and predict the codes of data items. This section describes the synthesis algorithm, along with several grouping and ranking algorithms used in PaTAT. We evaluate the performance of this synthesizer with an offline evaluation.

4.3.1 Synthesis algorithm. Unlike the "black-box" algorithms commonly used in other annotation domains, our algorithm generates interpretable patterns to inform and justify its recommendations (DG2). It uses inductive program synthesis based on user-provided positive and negative examples. At the time of synthesis, we use user annotated data to generate patterns that match a group of sentences. The patterns can be composed with `AND` (+) or `OR` (|) operators to become more expressive. Our pattern language consists of the following syntax:

- Part-of-speech (POS) tags: `VERB`, `PROPN`, `NOUN`, `ADJ`, `ADV`, `AUX`, `PRON`, `NUM`
- Word stemming: `[WORD]` (e.g., `[have]` will match all variants of have, such as *had*, *has*, and *having*)
- Soft match: `(word)` (e.g., `(pricey)` will match synonyms such as *expensive* and *costly*, etc.)
- Entity type: `$ENT-TYPE` (e.g., `$LOCATION` will match phrases of location type, such as *Houston, TX* and *California*; `$DATE` will match dates; `$ORG` will match names of organizations)
- Wildcard: * (will match any sequence of words)

The synthesizer has an initial search space described by the pattern syntax above. It iteratively traverses the search space using an enumerative search strategy, progressively assembling more complex patterns that better match user annotations. Given the data item annotations provided by users, our synthesizer generates patterns that match the positive examples and avoid matching the negative examples. During the synthesis process, a pattern will be abandoned if it is deemed not feasible; the feasibility of a pattern is measured by its reward (i.e., how many new true positive matches it introduces per unmatched positive examples), penalty (i.e., how many new false positive matches it introduces compared to its previous version), and depth (i.e., how many single patterns have been combined to create it). Through offline experiments, we found that a reward threshold of 0.01 and a penalty threshold of 0.3 produced a sufficiently thorough search (i.e., avoided abandoning promising patterns *too early*) without significantly prolonging search time. Typically the synthesizer produces a large number of rules and they may contain variations of each other (e.g., `NOUN`, `NOUN+*+ADJ`, `NOUN|PRON+ADV`). If a pattern's reward falls below the threshold, its penalty exceeds the threshold, or the reward falls below the penalty at its current state, the pattern will be pruned from the search.

Algorithm 1 Synthesis algorithm

```
while search space is not empty do
  for Every pattern in search space do
    Remove pattern from search space
    Create expandedPattern by combining pattern with previous pattern
    if expandedPattern is not feasible then
      Stop
    end if
    if expandedPattern expands positive match count then
      Expand pattern with +
      Expand pattern with |
    end if
    if PreviousPositive match count decreases or remains the same then
      Expand pattern with |
    end if
  end for
end while
```

For qualitative codes with rich semantic meanings, it is not feasible to capture their meanings with a single pattern. However, more complex and powerful ML models (e.g., neural networks and SVMs) pose interpretability challenges. To balance this trade-off, PaTAT uses a linear layer model to combine up to 5 individual patterns, picked through a pattern selection process, in order to make code recommendations for unlabeled data. The upper limit of 5 patterns was chosen through experimentation, balancing (1) the interpretability of the model's outcome, as it is infeasible for users to make sense of a large number of patterns, and (2) the expressive power of the model to capture codes with more complex meanings.

For the pattern selection process, PaTAT uses the variance of each pattern (determined by an ANOVA F-test [15]) to minimize overlaps in selected patterns. It starts with the pattern that has the highest

F1-score of all the candidate patterns, and then uses a greedy [26] approach to iteratively select patterns that would yield the maximum improvement in the overall F1-score. The selected patterns are used to train a linear model to predict whether a data item should be annotated with the current code. PaTAT also allows users to manually select patterns to be combined with other patterns through the linear layer, allowing users to promote patterns that they think are important and demote those that they consider irrelevant.

4.3.2 Grouping and ranking algorithms. To help users better understand the dataset and discover new themes while performing qualitative coding (Section 4.2.2), PaTAT supports several different grouping and ranking strategies:

- **Similarity Grouping:** Items with high semantic similarity are in the same group. We encode texts with a pre-trained language model (`all-MiniLM-L6-v2`[8]), and then generate groups using an agglomerative clustering method [45].
- **BERT-Model Prediction Grouping:** Items that are predicted to have the same code will be in the same group. We implement this grouping strategy as a multi-class classification task. With the current user annotations, we perform text classification with a pre-trained language model (`BERT-base-cased`[9]). As this strategy is based on user annotation, it is enabled only after the user starts to make data annotations.
- **Pattern-based Grouping:** This strategy groups items by the synthesized patterns that they match. Each of the currently synthesized patterns has a corresponding group. Each group contains items that match this pattern.
- **Most-Confident-Positive Ranking:** This ranking strategy ranks items by the confidence level of the model on a positive code annotation. Items that the model predicts are most likely to belong to the currently active code are at the top of the list. This strategy may help the user understand the current code by examining items that are considered "a clear fit" by the model.
- **Most-Confident-Negative Ranking:** Similarly, this ranking strategy ranks items by the model's confidence level on a negative code annotation. Items that the model predicts are least likely to belong to the currently active code are at the top of the list.
- **Least-Confident Ranking:** The items are ranked according to the confidence of the model, and items about which the model is least certain will be at the top of the list. This strategy may be useful if the user attempts to optimize the model's learning by providing ground-truth results for the model's most uncertain cases. It may also help the user review potential borderline items for opportunities to revise the codebook.

4.3.3 Offline performance evaluation. To understand the performance of our pattern synthesizer in capturing the patterns in text that predict codes and themes, we conducted an offline performance evaluation. In this evaluation, we compare our pattern synthesizer's accuracy of code predictions with that of a pre-trained BERT model [14].

Study Setup. In this experiment, we use two datasets:

- **YELP**: The YELP dataset[10] consists of user reviews of businesses (e.g., restaurants, retail stores) in 10 metropolitan areas in North America with 4 "ground-truth" categories, i.e., price, service, products, and environment.
- **ETHOS**: ETHOS [43] is a hate speech dataset that contains offensive comments from online platforms such as YouTube and Reddit with 8 "ground-truth" categories of offensive comments, i.e., violence, directed vs. generalized, gender, race, national origin, disability, religion, and sexual orientation.

To mimic the user's annotation process, we randomly sampled {1,2,3,4,5,10,15,20,25,...,115,120} items as training sets for each run to investigate how the prediction accuracy of each model changes as more training data (user annotations) becomes available. The prediction accuracy is always evaluated with the entire dataset to estimate how well each model captures the comprehensive meaning of each code. At each number of items, we repeated the training and evaluation process 10 times with different random seeds to obtain more reliable results.

Results. Figure 4 shows the results of the experiment. We observe that, except for BERT running on fewer than 5 user annotations where overfitting is most likely, the performance of both models generally improves as more data becomes available. The performance of PaTAT's model is comparable to that of BERT when trained on a smaller number of examples (fewer than 50–100). This represents the most common target use case of PaTAT, as the number of items labeled with each code tends to be small in most qualitative analysis tasks.

As the number of training data points grows, the neural-network-based BERT model starts to outperform PaTAT's pattern-based model, which is expected. However, we argue that our pattern-synthesis approach remains highly useful in many qualitative coding scenarios, where the advantages of transparency, interpretability, and user control enabled by a pattern-based method outweigh the improvement in accuracy that a "black-box" neural-network-based model would provide. It is worth noting that although the non-interactive *model-only* prediction accuracy of the pattern synthesizer might be lower than that of BERT with a large number of training examples, the pattern synthesizer can still be *more useful* than BERT in enabling accurate qualitative coding in human-AI collaborative settings because of the interface affordances patterns enable for user error discovery and repair (Fig. 3-3).

4.4 Implementation

4.4.1 Web application. The PaTAT front-end web application is implemented in React[11] and hosted using Python's built-in HTTP server. The back-end server is developed using the FastAPI[12] framework on the Firebase[13] platform that stores user annotations and log data. Both the front and back-end of PaTAT ran on a Google Cloud[14] server when we conducted the lab study.

[8]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[9]https://huggingface.co/bert-base-cased
[10]https://www.yelp.com/dataset
[11]https://reactjs.org/
[12]https://fastapi.tiangolo.com/
[13]https://firebase.google.com/
[14]https://cloud.google.com/

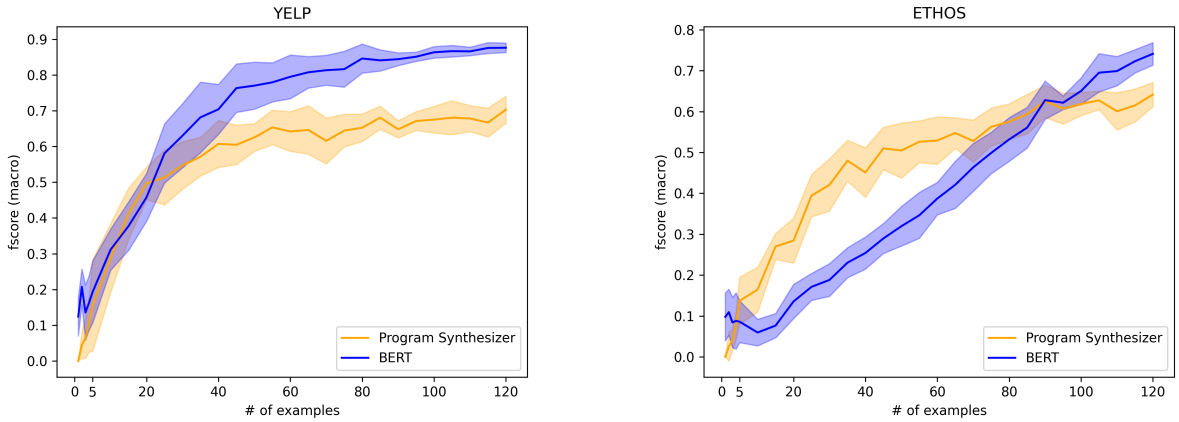

Figure 4: The change in prediction performance (F1-Score) of the BERT model and PaTAT's pattern synthesizer model over the number of training examples in YELP and ETHOS datasets. The shades represent the standard error of the mean F1-score.

4.4.2 Rule synthesis and matching. PaTAT uses the spaCy library to implement our synthesis algorithm (Section 4.3.1 and enable pattern matching. First, PaTAT uses spaCy for semantic parsing, predicting part of speech tags, and recognizing entities in a text. The tagging and entity information is then used by our synthesis algorithm in order to compose patterns complying with the common syntactic and semantic structures of labeled texts. PaTAT uses spaCy's Matcher to match patterns with texts for code recommendations and to look up the text portions that are matched by a pattern, enabling explanations.

5 USER STUDY

We conducted a lab user study with 8 participants with qualitative coding experience to evaluate PaTAT.[15] The study examined the following research questions:

- **RQ1**: How effective is PaTAT in providing explainable code recommendations for the user's annotation tasks in qualitative coding?
- **RQ2**: How useful is PaTAT for facilitating the learning of users, i.e., the discovery of new themes, trends, and connections among data items?
- **RQ3**: How well can PaTAT address the ambiguous, dynamic, and iterative nature of qualitative coding?

5.1 Participants

We used purposeful sampling [48] to recruit 8 participants (shown in Table 1) with experience in qualitative research. Of these 8 participants, two described themselves as having a "beginner level" of experience with Qualitative Data Analysis (QDA) (i.e., had done QDA fewer than four times), three had an intermediate level of QDA expertise (i.e., had done QDA four times in the last two years), and three had expert levels of expertise in QDA (i.e., had done QDA more than four times in the last two years). Regarding their background in machine learning (ML), two participants had no experience with ML, four participants had intermediate levels of expertise in ML (i.e., had taken an introductory ML course or understood ML theories), and two had expert levels of expertise in ML (i.e., very familiar with ML theories/extensive experience working in ML). Five participants had used some QDA assistance tools in the past, including NVIVO[16], Dedoose[17], Atlas.ti[18], and Miro[19], while the other three had not. Each participant was compensated with $60 USD for their time.

5.2 Study Design

Each study session lasted around 120 minutes and was conducted either in-person at a usability lab or virtually through Zoom. The study consisted of three qualitative coding task sessions with different conditional systems and a semi-structured interview (see Section 5.2.3 for procedure details).

An objective of the study design is to simulate the subjectivity, ambiguities, and uncertainties of qualitative coding. To achieve this, we started the study by introducing the dataset and a research question to the participants. Instead of providing participants with a predetermined list of labels, we asked them to come up with labels that could help them answer the research question, and we encouraged them to iterate on and continuously modify their codebook.

5.2.1 Datasets. We followed two criteria to select the datasets used in our user studies: (1) The dataset does not require any specialized domain knowledge to annotate; and (2) annotators can iteratively discover and develop hierarchical themes on the dataset. Based on these criteria, we selected the YELP and ETHOS dataset (Section 4). We randomly sampled 500 and 400 datapoints, respectively, and selected 100 datapoints at a time for the participants to work on.

5.2.2 Conditions. We used a within-subjects study design where each participant used the system under three different conditions:

[15]The study protocol was reviewed and approved by the IRB at the lead author's institution, where the study was conducted.

[16]https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home
[17]https://www.dedoose.com/
[18]https://www.atlasti.com/
[19]https://miro.com/

Participant ID	Level of education	Age	Gender	QDA expertise	QDA tools used	ML expertise
Par1	Master's	25-34	Female	Beginner	-	Expert
Par2	Master's	25-34	Female	Expert	Atlas.ti	Intermediate
Par3	Bachelor's	18-24	Male	Intermediate	Spreadsheet	Expert
Par4	Bachelor's	18-24	Female	Intermediate	R/Rstudio	Intermediate
Par5	Master's	18-24	Female	Intermediate	Nvivo, Dedoose	No experience
Par6	Master's	25-34	Non-binary	Expert	Atlas.ti, Saturate	Intermediate
Par7	Master's	18-24	Male	Expert	FigJam, Miro	Intermediate
Par8	Bachelor's	18-24	Male	Beginner	-	No experience

Table 1: Demographics of user study participants

- **MANUAL**: a baseline version of PaTAT without any intelligent features such as code recommendation, ranking, and grouping.
- **BERT**: a version of PaTAT using a "black-box" BERT model [14] to make code recommendations. Similar to the pattern synthesizer model, this BERT model also learns about the themes as the user annotates data and provides code recommendations for each item. However, no explanations (e.g., within-item text highlights, the display of learned patterns) were available due to the black-box nature of the model. Most ranking and grouping strategies were available, except for pattern-based grouping.
- **PaTAT**: a full version of PaTAT using our pattern-based synthesizer, with all key features including label recommendations, explanations, ranking, and grouping.

This approach allowed us to compare the performance, behavior, and experience of participants across all conditions, and to gather their feedback and insights about their interactions with PaTAT.

To control for potential biases, we randomly assigned each participant to one of the two datasets and used the same dataset in all three sessions for that participant. We randomly selected 100 data items from the selected dataset for each condition and ensured that the participants would not encounter the same item again in a different condition.

5.2.3 Study procedure. At the beginning of each study session, the experimenter collected informed consent and demographic information from the participant. Then the study coordinator gave a high-level explanation of the data as presented in Section 5.2.1. Following this, the participants were presented with a research question for which they would try to discover different themes in their assigned dataset (YELP or ETHOS). Each participant completed three sessions. Each session consisted of a 5–8 minute tutorial and a 30-minute qualitative coding task. The tutorial covered the key features of the tool assigned in the current condition. Before the qualitative coding task, the experimenter clarified that the goal of the task was to uncover interesting themes, annotate items with these themes, and learn about the dataset in the process of using the given tool. The experimenter also encouraged participants not to focus on only one theme but rather to create and collaborate with the model on a variety of themes. The themes created by the participant in each condition were not carried over into subsequent conditions. When the participant completed the three qualitative coding sessions, they filled out a post-study questionnaire. The session ended with a 10-minute semi-structured interview, where the experimenter asked questions to understand the participant's observed interesting behaviors during the tasks, how the participant used the tools, and the participant's user experience with the tool in each condition.

5.3 Study results

All participants interacted with a version of the system for 30 minutes in each condition. When interacting with the PaTAT condition, participants added 83 annotations with 8.5 codes on average. PaTAT achieves good prediction accuracy in code recommendations after learning from user annotations. The results also suggest that while the different interaction mechanisms used in PaTAT can introduce complexity, they make significant contributions to knowledge discovery, facilitating user learning, and overall user-perceived usefulness in the qualitative coding process.

5.3.1 Quantitative Results.

Prediction Accuracy. Since each participant was asked to create and iteratively develop their own codebook during the study, no global ground truth is available. Participants were able to select which of the themes they created they would focus on at any one time. Therefore, for each code that a participant created, we computed the precision, recall, and the F1-score of the code-specific model by comparing the predictions the model made with the participant's final set of annotations. For each participant, the aggregated precision, recall, and F1-scores reported in Table 2 are the macro-average [39] of each code-specific model's precision, recall, and F1-scores for all the codes created by the participant. A higher F1-score signifies a higher alignment of the user's mental model and the system's model. Table 2 also shows the number of model recommendation acceptances and rejections per participant, and the total number of annotations each participant added (manually or by accepting model recommendations). Overall, the results suggest that PaTAT achieves reasonably good prediction accuracy in code recommendations after learning from user annotations in real-time. This finding from the quantitative measures is corroborated by the results of our questionnaires (Section 5.3.1) and interviews (Section 5.3.2).

Post-Study Questionnaires. As shown in Table 3, the results of the post-study questionnaires show that MANUAL condition scored higher in the *ease of use* and *perceived user control* while the PaTAT condition scored higher in *usefulness* and *facilitating human learning of data*. Specifically, PaTAT was considered to be the most useful condition for annotation and the most effective condition for facilitating the user's learning of data characteristics by the

Par. ID	Dataset	Conditions	Total annotations	No. codes	Acc.	Rej.	Prec.	Recall	F1-score
Par1	YELP	P, M, B	82	8	29	29	0.69	0.58	0.57
Par2	YELP	B, M, P	65	18	19	19	0.92	0.75	0.79
Par3[20]	ETHOS	B, M, P	64	6	2	-	0.87	0.88	0.88
Par4	YELP	B, P, M	98	4	50	18	0.88	0.71	0.75
Par5	ETHOS	P, M, B	85	9	25	40	0.85	0.79	0.79
Par6	YELP	M, B, P	99	9	15	2	0.93	0.67	0.71
Par7	ETHOS	M, B, P	100	8	18	6	0.95	0.85	0.88
Par8	ETHOS	B, P, M	71	6	9	11	0.76	0.66	0.67
Average			**83**	**8.5**	**20.8**	**15.6**	**0.77**	**0.72**	**0.74**

[19] Par3 has few Acc. and Rej. because they interacted with PaTAT by annotating datapoints that the model had not previously labeled.

Table 2: The summary of participant interactions and model statistics in the PaTAT condition. *Total annotations* is the total number of annotated data items at the end of the 30 minute long PaTAT condition. Likewise, *No. codes* is the total number of codes identified by the participant, and *Acc* and *Rej* are the number of model-suggested codes that the participant accepted or rejected by the end of the PaTAT condition. The final prediction accuracy of the model in terms of precision, recall, and F1-score is captured in the final columns. As context, *Conditions* shows the order in which each participant experienced the Manual (M), BERT (B), and PaTAT (P) conditions.

Condition	Factor	1st Place	2nd Place	3rd Place
Manual	Ease of use	**6**	1	1
	Usefulness	0	2	6
	Perceived user control	**4**	3	1
	Facilitating human learning of data	0	4	4
BERT	Ease of use	1	3	4
	Usefulness	3	4	1
	Perceived user control	1	4	3
	Facilitating human learning of data	1	3	4
PaTAT	Ease of use	1	4	3
	Usefulness	**5**	2	1
	Perceived user control	3	1	4
	Facilitating human learning of data	**7**	1	0

Table 3: The summary of participant-reported rankings for three conditions on ease of use, usefulness, perceived user control, and effectiveness in facilitating their learning about data.

majority of participants across all three conditions. Participants considered PaTAT more difficult to use and less controllable relative to fully manual annotation, but, compared to the other AI-assisted approach—the "black box" BERT condition—PaTAT's interpretable pattern-based approach shows a clear advantage in usefulness, user-perceived sense of control, and facilitation of human learning.

5.3.2 Post-Study Interviews. Following established open-coding guidelines [5, 31], two members of the study team first individually conducted a round of qualitative coding. They then discussed the codes together to reach a consensus and develop a unified codebook. Subsequently, they conducted a thematic analysis using the unified codebook to identify emerging themes raised in the interview. The entire research team examined the results of the coding process together to discover higher-level themes. We report the following key findings:

KF1: The role of AI assistance can be active or passive in collaboration with users, depending on user contexts and preferences. Participants naturally adopted different annotation practices on their own as they interacted with PaTAT. For example, Par5 rejected a large number of code recommendations, despite the fact that the F1 score of the model is reasonably high (Table 2). This was due to Par5 intentionally looking for "wrong predictions" in an effort to correct what the model has learned. Some participants preferred to use the system passively, where they led the annotation process and only used the suggestions of the model as a way to validate their manual annotations. In this paradigm, participants did not annotate with the explicit intention of helping the model learn, resulting in less interaction with the model's predictions. For example, Par3 spent most of their efforts on data items that had not been annotated by PaTAT. Passive AI assistance was perceived by participants as having positive effects on coding efficiency: *"It can save time, too, because sometimes I have to do multiple passes. But I can see here it is already doing that for me. So I do not have to put in as much effort of trying to validate my coding."* (Par6)

On the other hand, some participants preferred to have the AI model play a more active role, where they relied on the model suggestions to guide their annotation process and expand the codebook after "understanding what it [PaTAT] was doing" (Par1). These

participants used two strategies to incorporate the model recommendations into their annotating process: (1) explicitly accepting or rejecting the model's suggestions (Par4, Par5) and (2) trying to identify data points the model was not able to capture while using the generated patterns to keep track of the codes and be *"more consistent"* (Par3) in their annotation. While participants had different approaches to their collaboration with PaTAT, they saw value of it in allowing them to "go through annotation quickly" (Par5) or "capturing things that might have been missed" (Par6).

KF2: Understanding the current state of the model contributes to error discovery and repair. Participants found the user-interpretable patterns synthesized by PaTAT to be valuable. Some participants used them as a way to validate the capability of the system: *"For example, if it is coding for 'good price', I am looking for a positive adjective plus a synonym for price or discounts or whatever. So when I see that it is on the right track and doing something like that, it makes me a lot more confident in its ability to predict, as well as its ability to capture things that I might have missed if I were the only one going through the data"* (Par6). Similarly, the participants actively attempted to correct the model by accepting or rejecting model suggestions.

The benefit of enabling participants to directly view and modify model-learned patterns was evident in the approach they took when performing repairs, but participants also pointed out that this came with additional cognitive load relative to working with a fully manual tool: *"So I think it is really helpful to know why it is predicting. But I think, for a first pass through, I would want to go through and do a sort of overview and then get into the granular details"* (Par5). Further into their interview, Par5 found different levels of granularity of annotation (i.e., highlighting and interacting with portions of text within a data item) useful for model repair, especially when the model "was right for the wrong reason": *"It seemed to be doing somewhat of a decent job of clustering alike examples... But then what it actually highlighted words like, 'seems like', or 'seen' ...So I kept trying to re-highlight something else, and it seemed to be getting better pretty quickly [as a result]."* The different granularities of annotation contributed to the participants' perceived levels of control in validating and repairing what the model should match: *"I like the amount of control I had over being able to say, Okay, when I say 'girl', I want it to be 'girl' and 'woman' and 'son' and 'daughter', and not 'kid' or 'child.'"* (Par5)

KF3: The abstractness of the code and the data associated with it affects PaTAT's ability to assist the user. Due to their abstract nature, certain themes that participants attempted to capture in a code, e.g., 'atmosphere' or 'discrimination', were more challenging for PaTAT's model to accurately capture from a few examples. In addition, the PaTAT model's difficulty in capturing these abstract codes sometimes resulted in the generation of empty sets, where the model was unable to identify any patterns that matched the desired code to the data items the user attached it to. This resulted in a lower aggregate F1-score when evaluated with the participant's final annotations as a ground truth. These cases were characterized by annotations with a broad scope of examples containing subtle and non-symbolic distinctions. For example, Par4 spent most of their time solely working on the code "atmosphere" while other participants switched between different codes, balancing out the model's learning for each code. When the model did not do a particularly good job of capturing patterns for the current code from the participant's annotations provided so far, some participants adopted a *'confirmation'* (Par1) strategy for interacting with the tool: *"A quick glance at the highlighted word, and it was easier to select sentences the fit the theme."* (Par4)

KF4: Interpretable patterns facilitate the discovery of new themes, trends, and connections among data items. Patterns that captured outliers in a dataset helped participants notice important details and nuanced aspects of the data. Participants reported that they noticed the model picking up patterns describing what they were keeping in mind themselves as they annotated the data: *"In some of them [patterns are] very simple. But in practice that is how I code a sentence. So if the pattern can detect this simple word combination, it will be very useful for me"* (Par2). Participants reported that the synthesized patterns are also helpful for justifying their annotation: *"I am not too sure how I wound report it? I cannot say the model picked up this pattern and the data that we labeled, we probably still have to do manual processing to say like eighty percent of the tweets that are racial related had the word police in it, or something like that would make the report more rigorous."* (Par7)

Participants reported that they learned higher-level trends within and meta-data about the dataset from the synthesized patterns, which helped them form new theories and extract new insights from the data. Par5 discussed their experience of seeing different words appear together in patterns, which led to an insightful discovery: *"Okay, am I seeing ideas around women coming up with ideas around violence more? Or they had patterns with different words, like language around body parts and stuff like that [...] If I am looking at gender, how do those match up with other themes and other ideas? For example, if I am interested in gender and sexuality, if I am seeing actually more specific language about body parts when we are talking about gender than we are talking about sexuality, I could see that being useful."* (Par5)

KF5: Different rankings and groupings of data accommodate the ambiguous, dynamic, and iterative nature of qualitative coding. Participants frequently experimented with different grouping strategies during sessions. These interactions helped participants review and revise their codes. The ability to find *"well-chained themes"* (Par1) and similar sentences helped participants find higher-level themes *"instead of translating the sentence into a label by looking at each sentence"* (Par2). Participants reported having to put in less effort to interpret "similarity between sentences rather than labels" (Par2). Participants also used these features as a way to iteratively incorporate the model into their own work. The most-used ranking strategy was ranking by the most confident positive annotations. The participants used the confidence ranking to decide to "accept" the suggestions of the model or "wait another iteration of retraining" (Par5). As participants reached data items where the confidence of the model was low, there were cases where borderline suggestions were helpful in "identifying tricky cases" (Par1) to correct the model. Some participants found it harder to accept model suggestions after going through the top most confident suggestions, and abandoned the ranking setting.

6 DISCUSSION

6.1 The Impact of of AI Assistance on User Workflows in Qualitative Coding

Generally, researchers design codes to capture concepts in the data [22]. The coding process is accompanied by a sparsity in the positive-labeled data due to its iterative and dynamic nature. Within a human-AI collaboration, this leads us to rethink the common wisdom regarding the optimization of recall and precision of selected patterns as the "ultimate goal" during the formulation of codebook. Prior work by Chen et al. [6] suggests that a way to bridge this gap is to make model building a meaningful task in the researcher's workflow. The results of our study show that, by taking advantage of the strengths of the human-AI team, researchers can act on model-generated feedback and perform ongoing refinement and evolution of the coding framework with the assistance of the system. This can help streamline and enhance the coding process, allowing researchers to effectively analyze and interpret their data.

Given the subtleties and nuances that can exist within text, particularly when the themes being coded are abstract in nature, PaTAT introduces soft matches within the pattern language that provide more flexibility in the matching process and capture a wider latent space. The patterns are combined through a linear layer to further expand the match coverage while still ensuring the model's explainability and users' ability to validate and correct it. These observations highlight the need for pattern languages that can capture more abstract meanings of codes in the design of AI assistance for qualitative coding.

We believe that there is a trade-off between supporting abstraction and the ability to enable human reasoning and explanability for qualitative coding. As illustrated in Fig. 5, PaTAT has made significant progress toward supporting codes with higher levels of abstraction and complexity by introducing *soft matches* and combinations of patterns through a *linear layer* while maintaining the interpretability of the pattern rules. There are future research opportunities to understand and explore this design space, expanding the expressiveness of the model to capture more abstract and complex themes while retaining the interpretability needed to support user control, validation, and human learning about the data.

6.2 Different Stages of the Coding Process Call for Different Forms of Assistance

Different stages of the coding process, as portrayed by the user's familiarity with the data and the formation of the codebook, call for different forms of assistance from the system [18, 41]. A common challenge in the early stage of qualitative coding is the dynamic formation of codebooks and the evolving interpretation of codes by researchers [41]. Aiming for coder agreement (in our case, human-AI agreement) too early can be undesirable or even counterproductive when developing the initial codebook [42]. Our participants cited different features of PaTAT for the roles they play at different stages of the coding process (KF1–KF5). Participants saw the value of PaTAT's code suggestions after coming up with some initial codes themselves. Before then, participants indicated that they wanted to have autonomy that resembles manual annotation with assistance primarily in data organization to facilitate their understanding of the data (KF2). This stage of analysis is mainly driven by the participants' own interpretations of the data. Following the formation of the initial codebook, participants started to use PaTAT's suggestions to ensure coding consistency, as well as to expand the codebook and the researcher's interpretation of the data. These findings motivate future research on adaptive interaction strategies that adjust the levels and forms of AI assistance depending on the current stage of the coding process.

6.3 Working with Ambiguities and Uncertainties

A unique characteristic of qualitative coding as a human-AI collaboration task is the inherent ambiguities and uncertainties during the annotation process [27]. This characteristic makes it more complicated for AI to continually learn from human annotations, since annotators could iteratively modify themes and codes, making human-annotated labels themselves unstable. To avoid negative feedback loops and establish trustworthy human-AI partnerships in convoluted settings, users must understand (1) what the model has learned and (2) why the model makes a certain recommendation to effectively validate and repair model results [50].

The design of PaTAT demonstrates three promising strategies to help users explicitly disambiguate model uncertainties: first, PaTAT provides users with group-wise annotation features that allow them to specify a search space of overall patterns for a code that applies to a batch of data items. Second, the user can direct the AI model's attention to certain regions of a data item to specify not only the correct result, but also which relevant part of the data leads to the result. Lastly, PaTAT allows users to select and directly adjust the patterns used for coding recommendations, reducing uncertainty in the pattern selection phase. These strategies exemplify how users can contribute to disambiguation at different stages of using an AI-enabled tool that is learning from them.

Since codes can be inherently abstract and complex, it can be difficult for the model to identify specific themes or connotations that a group of concrete data items share. For instance, the model may learn different combinations of patterns, but not necessarily the metaphorical meanings that may be present in the data. In these cases, it becomes crucial for users to reflect on what the model as learned so far and identify any areas that may still require particular attention and deeper analysis, highlighting the importance of supporting effective error discovery and repair in human-AI collaboration [35, 47].

6.4 Coordinating Model Learning and Human Learning

Human-AI collaborative data annotation tools in most task domains focus primarily on optimizing for model learning. However, in domains such as qualitative coding, scaffolding human learning is just as or more important. The design of PaTAT demonstrates strategies to accommodate goals that may be generalizable to other human-AI collaboration task domains where the continued evolution of the user's knowledge is important, e.g., human-AI co-creation, content moderation, and intelligent tutors.

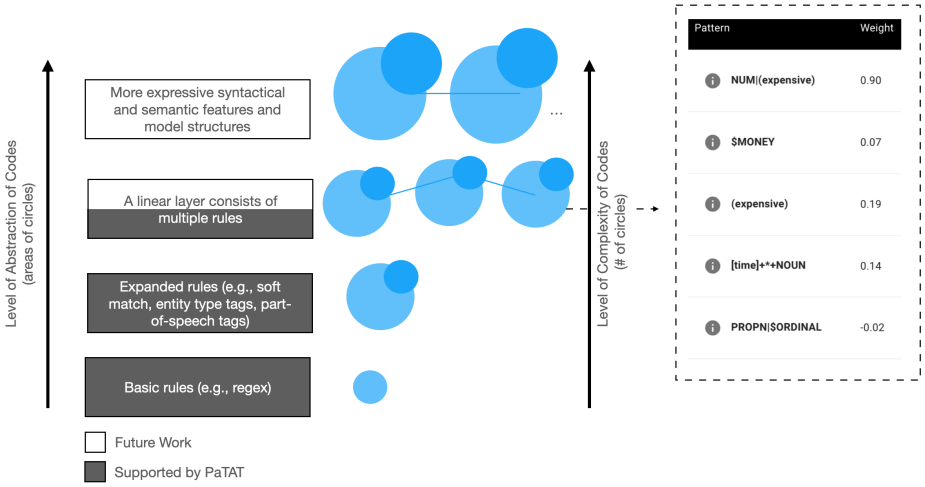

Figure 5: An illustration that the required expressiveness of the pattern language rises as the desired codes get more abstract (the circles that indicate the meanings of the code get bigger) and complex (the increasing numbers of circles indicate that the meanings of the code get more ambiguous or multifaceted). Our work demonstrates the need for supporting higher-level abstractions for AI-enabled support in qualitative coding. PaTAT makes several strides in this direction by introducing features such as *softmatch* and design choices such as *linear combinations of small numbers of highly expressive patterns*. We leave for future work the next steps in expanding the expressiveness of the pattern language while retaining interpretability and user control.

At the interaction mechanism level, instead of presenting data in an order that optimizes for model learning (e.g., showing the items that the model is most uncertain about), PaTAT supports presenting data orders that the user finds most helpful to them, supporting their understanding of global data trends, local semantic clusters, or outliers in a group. At the level of data representations, existing tools tend to use "black-box" learning representations, such as embeddings, to maximize the model's prediction accuracy. In contrast, PaTAT synthesized interpretable patterns to represent what the model has learned as an attempt to inspire users' reflection on their own "mental schema" for annotation making, as well as help them discover new potentially meaningful patterns that they were not yet aware of. Lastly, at the level of interface features, PaTAT provides interactive highlighting and exploration features on the data that help users understand the current state of the model with a focus on how the model applies the learned rules to particular data items.

One feature in PaTAT, i.e., grouping data items into semantically-similar clusters, used "black box" models. Interestingly, users did not seem to be troubled as much by the lack of explanation there as the lack of explanation in the code recommendations in the BERT condition of the study. A possible explanation of this phenomenon is that the grouping feature does not automate the end-to-end process of the main task (i.e., annotation) but instead automates a sub-task (data organization) that contributes to the main task. The concept of semantic similarity in the black box model used for grouping was also relatively straightforward compared to end-to-end code recommendations. We hope that these strategies can inspire the design of future human-AI collaborative tools to manage the balance between the learning of the model and the learning of human users.

7 LIMITATIONS AND FUTURE WORK

The current version of PaTAT presents several technical limitations. The syntax of our pattern language is already more expressive and flexible than previous rule-based qualitative coding assistance systems such as Cody [52] with support for entity types (e.g., `$DATE`, `$ORG`, and `$EVENT`) and synonyms, but PaTAT's syntax does not yet represent more expressive natural language features, such as sentiment, co-references, and commonsense knowledge. To accommodate the larger search space that would come with the expansion of the pattern language, more effective synthesis and pruning algorithms are needed, in order to cover the search space more efficiently and/or shrink the search space during pattern synthesis. This would reduce the computation time for each retraining round while the user continues to annotate data.

The current PaTAT interface supports the use of only one user at a time, while qualitative coding is often a collaborative effort with multiple users involved. With the current version of PaTAT, multiple users can first use PaTAT to annotate data individually and then come together (without AI assistance) to discuss their coding results to reach a consensus and discover additional insights as a group. This presents an opportunity for future work that helps facilitate this collaborative process. A new system can, for example, help identify conflicts and discrepancies in coding results, reveal potential biases in the coding process, assemble a group of annotators to minimize biases, and facilitate conflict resolutions.

PaTAT offers many interface features and strategies to support ranking, grouping, interpreting, exploring, and validating data items and the model's recommendations. Although all of them are useful in *some* contexts, the findings of our user study suggest

that most of them are most useful in *different* contexts. The large number of features also contributes to the complexity of the interface, affecting the ease of use of the system. Moving forward, a promising direction for future research is to make PaTAT more adaptive, so that it can recommend the most useful interface features and strategies for the current task domain, the use context, the analysis goals, and user preferences.

Following our lab study, we plan to conduct a larger-scale deployment study and eventually a public release of PaTAT. We believe that this will not only allow a broad group of qualitative researchers to take advantage of PaTAT's AI assistance in their own task domains, but also enable us to study the long-term in-situ usage of PaTAT in its intended context of use. We hope that this future study can evaluate PaTAT's real-world usefulness, validate its ecological validity, uncover insights into how users adopt PaTAT in different task domains, and identify future research opportunities.

Finally, a limitation of our lab study is its lack of quantitative results on how the use of PaTAT impacts the efficiency of the qualitative coding process, despite the fact that the qualitative findings suggest that users perceive the use of PaTAT as having positive effects on their coding efficiency (KF1 in Section 5.3.2). Although Table 2 reports the number of annotations each participant made in a study session under different conditions, we believe that it is not an appropriate measure for the efficiency of the qualitative coding process. As we have discussed in earlier sections, the goal of qualitative coding is more than just annotating all data items, but allowing the user to understand the data, come up with and continuously evolve the codebook, and discover patterns *through* the coding process. The planned deployment study will allow us to explore this aspect during real-world usage of PaTAT.

8 CONCLUSION

We presented PaTAT, a human-AI collaborative tool that assists users with qualitative coding. PaTAT uses a new explainable interactive pattern synthesis approach that (1) learns about the codes as users annotate more data, (2) provides code recommendations, (3) explains what the model has learned, (4) accommodates the development and evolution of codebooks, and (5) facilitates users' learning of data characteristics, trends, patterns in order for them to form new theories and insights. A user study with 8 qualitative researchers illustrated PaTAT's usefulness in providing qualitative coding assistance and effectiveness in facilitating the human learning of data. This work also presents design implications for human-AI collaboration when working with ambiguities and uncertainties, coordinating model learning and human learning, and supporting user understanding of model status and rationales.

ACKNOWLEDGMENTS

This work was supported in part by an AnalytiXIN Faculty Fellowship, an NVIDIA Academic Hardware Grant, a Google Cloud Research Credit Award, a Google Research Scholar Award, and the NSF Grants 2211428, 2107391 and 2123965. Any opinions, findings or recommendations expressed here are those of the authors and do not necessarily reflect views of the sponsors.

REFERENCES

[1] John Ahlgren and Shiu Yin Yuen. 2013. Efficient program synthesis using constraint satisfaction in inductive logic programming. *The Journal of Machine Learning Research* 14, 1 (2013), 3649–3682.
[2] Tehmina Basit. 2003. Manual or electronic? The role of coding in qualitative data analysis. *Educational research* 45, 2 (2003), 143–154.
[3] Pat Bazeley. 2009. Analysing qualitative data: More than 'identifying themes'. *Malaysian Journal of Qualitative Research* 2, 2 (2009), 6–22.
[4] Richard E Boyatzis. 1998. *Transforming qualitative information: Thematic analysis and code development*. Sage.
[5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
[6] Nan-Chen Chen, Margaret Drouhard, Rafal Kocielnik, Jina Suh, and Cecilia R Aragon. 2018. Using machine learning to support qualitative coding in social science: Shifting the focus to ambiguity. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8, 2 (2018), 1–20.
[7] Nan-chen Chen, Rafal Kocielnik, Margaret Drouhard, Vanessa Peña-Araya, Jina Suh, Keting Cen, Xiangyi Zheng, and Cecilia R Aragon. 2016. Challenges of applying machine learning to qualitative coding. In *ACM SIGCHI Workshop on Human-Centered Machine Learning*.
[8] Xinyun Chen, Chang Liu, and Dawn Song. 2017. Towards synthesizing complex programs from input-output examples. *arXiv preprint arXiv:1706.01284* (2017).
[9] Dirk Colbry, Fred Dyer, Ian Dworkin, Yang Wang, and Lifeng Wangs. 2013. Speeding up scientific imaging workflows: Design of automated image annotation tool. In *2013 1st IEEE Workshop on User-Centered Computer Vision (UCCV)*. IEEE, 13–18.
[10] Kevin Crowston, Eileen E Allen, and Robert Heckman. 2012. Using natural language processing technology for qualitative data analysis. *International Journal of Social Research Methodology* 15, 6 (2012), 523–543.
[11] Kevin Crowston, Xiaozhong Liu, and Eileen E Allen. 2010. Machine learning and rule-based automated coding of qualitative data. *proceedings of the American Society for Information Science and Technology* 47, 1 (2010), 1–2.
[12] Katherine Darveau, Daniel Hannon, and Chad Foster. 2020. A Comparison of Rule-Based and Machine Learning Models for Classification of Human Factors Aviation Safety Event Reports. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 64. SAGE Publications Sage CA: Los Angeles, CA, 129–133.
[13] Cristina David and Daniel Kroening. 2017. Program synthesis: challenges and opportunities. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 375, 2104 (2017), 20150403.
[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[15] Nadir Omer Fadl Elssied, Othman Ibrahim, and Ahmed Hamza Osman. 2014. A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology* 7, 3 (2014), 625–638.
[16] Jessica L. Feuston and Jed R. Brubaker. 2021. Putting Tools in Their Place: The Role of Time and Perspective in Human-AI Collaboration for Qualitative Analysis. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 469 (oct 2021), 25 pages. https://doi.org/10.1145/3479856
[17] Pierre Flener. 2002. Achievements and prospects of program synthesis. *Computational logic: logic programming and beyond* (2002), 310–346.
[18] Abbas Ganji, Mania Orand, and David W McDonald. 2018. Ease on Down the Code: Complex Collaborative Qualitative Coding Simplified with'Code Wizard'. *Proceedings of the ACM on human-computer interaction* 2, CSCW (2018), 1–24.
[19] Lindsay Giesen and Allison Roeser. 2020. Structuring a team-based approach to coding qualitative data. *International Journal of Qualitative Methods* 19 (2020), 1609406920968700.
[20] Monica M Gonzalez. 2016. The coding manual for qualitative research: A review. *The Qualitative Report* 21, 8 (2016), 1546–1549.
[21] Mohamed Goudjil, Mouloud Koudil, Mouldi Bedda, and Noureddine Ghoggali. 2018. A novel active learning method using SVM for text classification. *International Journal of Automation and Computing* 15, 3 (2018), 290–298.
[22] Stephen Gough and William Scott. 2000. Exploring the purposes of qualitative data coding in educational enquiry: Insights from recent research. *Educational Studies* 26, 3 (2000), 339–354.
[23] Timothy C Guetterman, Tammy Chang, Melissa DeJonckheere, Tanmay Basu, Elizabeth Scruggs, and VG Vinod Vydiswaran. 2018. Augmenting qualitative text analysis with natural language processing: methodological study. *Journal of medical Internet research* 20, 6 (2018), e9702.
[24] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices* 46, 1 (2011), 317–330.
[25] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. 2017. Program synthesis. *Foundations and Trends® in Programming Languages* 4, 1-2 (2017), 1–119.
[26] Dorit S Hochbaum and Anu Pathria. 1998. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)* 45, 6 (1998),

615–627.
[27] Jialun Aaron Jiang, Kandrea Wade, Casey Fiesler, and Jed R. Brubaker. 2021. Supporting Serendipity: Opportunities and Challenges for Human-AI Collaboration in Qualitative Analysis. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 94 (apr 2021), 23 pages. https://doi.org/10.1145/3449168
[28] Holtzblatt Karen and Jones Sandra. 2017. Contextual inquiry: A participatory technique for system design. In *Participatory design*. CRC Press, 177–210.
[29] Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*. 5–9.
[30] Kanako Komiya, Masaya Suzuki, Tomoya Iwakura, Minoru Sasaki, and Hiroyuki Shinnou. 2018. Comparison of methods to annotate named entity corpora. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 17, 4 (2018), 1–16.
[31] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research methods in human-computer interaction*. Morgan Kaufmann, Chichester, West Sussex, U.K.
[32] Vu Le, Daniel Perelman, Oleksandr Polozov, Mohammad Raza, Abhishek Udupa, and Sumit Gulwani. 2017. Interactive program synthesis. *arXiv preprint arXiv:1703.03539* (2017).
[33] Woosuk Lee, Kihong Heo, Rajeev Alur, and Mayur Naik. 2018. Accelerating search-based program synthesis using learned probabilistic models. *ACM SIGPLAN Notices* 53, 4 (2018), 436–449.
[34] Laurent Letourneau-Guillon, David Camirand, Francois Guilbert, and Reza Forghani. 2020. Artificial intelligence applications for workflow, process optimization and predictive analytics. *Neuroimaging Clinics* 30, 4 (2020), e1–e15.
[35] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M. Mitchell, and Brad A. Myers. 2020. Multi-Modal Repairs of Conversational Breakdowns in Task-Oriented Dialogs. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST 2020)*. ACM. https://doi.org/10.1145/3379337.3415820
[36] Zhuofan Li, Daniel Dohan, and Corey M Abramson. 2021. Qualitative Coding in the Computational Era: A Hybrid Approach to Improve Reliability and Reduce Effort for Coding Ethnographic Interviews. *Socius* 7 (2021), 23780231211062345.
[37] 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*. 44–48.
[38] Kathleen M MacQueen, Eleanor McLellan, Kelly Kay, and Bobby Milstein. 1998. Codebook development for team-based qualitative analysis. *Cam Journal* 10, 2 (1998), 31–36.
[39] Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.
[40] Megh Marathe and Kentaro Toyama. 2018. Semi-Automated Coding for Qualitative Research: A User-Centered Inquiry and Initial Prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173922
[41] Megh Marathe and Kentaro Toyama. 2018. Semi-automated coding for qualitative research: A user-centered inquiry and initial prototypes. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–12.
[42] 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 (2019), 1–23.
[43] Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2020. ETHOS: an online hate speech detection dataset. *arXiv preprint arXiv:2006.08328* (2020).
[44] Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. 2019. A BERT-based transfer learning approach for hate speech detection in online social media. In *International Conference on Complex Networks and Their Applications*. Springer, 928–940.
[45] Fionn Murtagh and Pedro Contreras. 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, 1 (2012), 86–97.
[46] Laura K Nelson. 2020. Computational grounded theory: A methodological framework. *Sociological Methods & Research* 49, 1 (2020), 3–42.
[47] Zheng Ning, Zheng Zhang, Tianyi Sun, Yuan Tian, Tianyi Zhang, and Toby Jia-Jun Li. 2023. An Empirical Study of Model Errors and User Error Discovery and Repair Strategies in Natural Language Database Queries. In *Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI '23)*.
[48] Lawrence A Palinkas, Sarah M Horwitz, Carla A Green, Jennifer P Wisdom, Naihua Duan, and Kimberly Hoagwood. 2015. Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. *Administration and policy in mental health and mental health services research* 42, 5 (2015), 533–544.
[49] Keith F Punch. 2013. *Introduction to social research: Quantitative and qualitative approaches*. sage.
[50] Sarvapali D Ramchurn, Sebastian Stein, and Nicholas R Jennings. 2021. Trustworthy human-AI partnerships. *Iscience* 24, 8 (2021), 102891.
[51] Tim Rietz and Alexander Maedche. 2020. Towards the Design of an Interactive Machine Learning System for Qualitative Coding.. In *ICIS*.
[52] 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* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 394, 14 pages. https://doi.org/10.1145/3411764.3445591
[53] Cynthia Rudin. 2015. Can machine learning be useful for social science. *The Cities: An essay collection from the Decent City initiative* 9 (2015), 86–90.
[54] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial sketching for finite programs. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*. 404–415.
[55] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. 102–107.
[56] Anselm Strauss and Juliet Corbin. 1990. *Basics of qualitative research*. Sage publications.
[57] Anselm L Strauss. 1987. *Qualitative analysis for social scientists*. Cambridge University Press.
[58] Chen Sun, Jean M Uwabeza Vianney, Ying Li, Long Chen, Li Li, Fei-Yue Wang, Amir Khajepour, and Dongpu Cao. 2020. Proximity based automatic data annotation for autonomous driving. *IEEE/CAA Journal of Automatica Sinica* 7, 2 (2020), 395–404.
[59] Gerhard B Van Huyssteen and Martin J Puttkammer. 2007. Accelerating the annotation of lexical data for less-resourced languages.. In *INTERSPEECH*. 1505–1508.
[60] David Wicks. 2017. The coding manual for qualitative researchers. *Qualitative research in organizations and management: an international journal* (2017).
[61] Charlene A Winters, Shirley Cudney, and Therese Sullivan. 2010. The Evolution of a Coding Schema in a Paced Program of Research. *Qualitative Report* 15, 6 (2010), 1415–1430.
[62] Megan Woods, Trena Paulus, David P Atkins, and Rob Macklin. 2016. Advancing qualitative research using qualitative data analysis software (QDAS)? Reviewing potential versus practice in published studies using ATLAS. ti and NVivo, 1994–2013. *Social Science Computer Review* 34, 5 (2016), 597–617.
[63] Tongshuang Wu, Kanit Wongsuphasawat, Donghao Ren, Kayur Patel, and Chris DuBois. 2020. Tempura: Query analysis with structural templates. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.
[64] Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*. 2159–2169.
[65] Seid Muhie Yimam, Chris Biemann, Richard Eckart de Castilho, and Iryna Gurevych. 2014. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, 91–96. https://doi.org/10.3115/v1/P14-5016
[66] Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 1–6.
[67] Ruixue Zhang, Wei Yang, Luyun Lin, Zhengkai Tu, Yuqing Xie, Zihang Fu, Yuhao Xie, Luchen Tan, Kun Xiong, and Jimmy Lin. 2020. Rapid adaptation of bert for information extraction on domain-specific business documents. *arXiv preprint arXiv:2002.01861* (2020).
[68] Tianyi Zhang, London Lowmanstone, Xinyu Wang, and Elena L Glassman. 2020. Interactive program synthesis by augmented examples. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 627–648.
[69] Jingbo Zhu, Huizhen Wang, Benjamin K Tsou, and Matthew Ma. 2009. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on audio, speech, and language processing* 18, 6 (2009), 1323–1331.
[70] Sue Ziebland and Ann McPherson. 2006. Making sense of qualitative data analysis: an introduction with illustrations from DIPEx (personal experiences of health and illness). *Medical education* 40, 5 (2006), 405–414.