SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Tool Support for Human-AI Collaboration in Qualitative Data Analysis

Submitted by

Jie GAO

Thesis Advisors

Dr. Simon PERRAULT

Information Systems Technology and Design Pillar

A thesis submitted to the Singapore University of Technology and Design in fulfilment of the requirement for the degree of Doctor of Philosophy

2024

PhD Thesis Examination Committee

Advisor:	Dr. Simon PERRAULT
TEC Chair:	Prof. LU Wei
Internal TEC member 1:	Dr. Dorien HERREMAN
Internal TEC member 2:	Dr. LIM Kwan Hui

Abstract

Information Systems Technology and Design Pillar

Doctor of Philosophy

Tool Support for Human-AI Collaboration in Qualitative Data Analysis

by Jie GAO

Qualitative data analysis, a crucial subset of data analysis, extracts insights from unstructured data through systematic methods. It plays a significant role in diverse fields such as social science, psychology, and educational research. With the rapid evolution of Artificial Intelligence (AI) and Large Language Models (LLMs), leveraging AI, particularly LLMs, in qualitative analysis has gained considerable attention. The synergy between LLMs and human analysts in producing in-depth interpretations is becoming more and more important.

This PhD thesis, initiated before the advent of LLMs, delves into the then-niche area of AI-assisted qualitative analysis. It investigates the potential of AI in augmenting collaborative qualitative analysis, focusing on human-to-human and direct human-AI collaboration. The thesis is structured into two main sections: 1) developing innovative collaborative workflows for qualitative coding teams, integrating multiple humans and AI models, and 2) enhancing trust and reliance dynamics within human-AI teams by analyzing their interactions.

More specifically, I will begin by setting the context and motivations for this burgeoning field, followed by an overview of the core theoretical frameworks in qualitative analysis. Subsequently, I will introduce the design and development of two CQA systems, CoAIcoder and CollabCoder, along with the associated studies and evaluations. Lastly, I will present an empirical study examining the trust and reliance dynamics between humans and AI in a qualitative analysis context.

Publications

In reverse chronological order:

1. **Jie Gao**, Yuchen Guo, Gionnieve Lim, Tianqin Zhang, Zheng Zhang, Toby Jia-Jun Li, and Simon Tangi Perrault. 2023. CollabCoder: A Lower-barrier, Rigorous Workflow for Inductive Collaborative Qualitative Analysis with Large Language Models. ACM CHI conference on Human Factors in Computing Systems (CHI'24). https://doi.org/10.48550/arXiv.2304.07366

2. **Jie Gao**, Kenny Tsu Wei Choo, Junming Cao, Roy Ka-Wei Lee, and Simon Perrault. 2023. CoAIcoder: Examining the Effectiveness of AI-assisted Human-to-Human Collaboration in Qualitative Analysis. ACM Transactions on Computer-Human Interaction (TOCHI) 31.1 (2023): 1-38. https://doi.org/10.1145/3617362

3. **Jie Gao**, Yuchen Guo, Toby Jia-Jun Li, and Simon Tangi Perrault. 2023. CollabCoder: A GPT-Powered WorkFlow for Collaborative Qualitative Analysis. In Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing (CSCW '23 Companion). Association for Computing Machinery, New York, NY, USA, 354–357. https://doi.org/10.1145/3584931.3607500

4. **Jie Gao**, Junming Cao, ShunYi Yeo, Kenny Tsu Wei Choo, Zheng Zhang, Toby Jia-Jun Li, Shengdong Zhao, and Simon Tangi Perrault. 2023. Impact of Human-AI Interaction on User Trust and Reliance in AI-Assisted Qualitative Coding. https://doi.org/10.48550/arXiv.2309.13858 **(Under review)**

5. Zheng Zhang, **Jie Gao**, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023. VISAR: A Human-AI Argumentative Writing Assistant with Visual Programming and Rapid Draft Prototyping. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23). Association for Computing Machinery, New York, NY, USA, Article 5, 1–30. https://doi.org/10.1145/3586183.3606800

6. Nuwan Janaka, **Jie Gao**, Lin Zhu, Shengdong Zhao, Lan Lyu, Peisen Xu, Maximilian Nabokow, Silang Wang, and Yanch Ong. 2023. GlassMessaging: Towards Ubiquitous Messaging Using OHMDs. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT'23). 7, 3, Article 100 (September 2023), 32 pages. https://doi.org/10.1145/3610931

7. Junming Cao and Bihuan Chen and Longjie Hu and **Jie Gao** and Kaifeng Huang and Xin Peng. 2023. Characterizing the Complexity and Its Impact on Testing in ML-Enabled Systems - A Case Study on Rasa. 2023 IEEE International Conference

on Software Maintenance and Evolution (ICSME'23), Bogota, Colombia. https://doi.org/10.1109/ICSME58846.2023.00034

8. **Jie Gao**, Xiayin Ying, Junming Cao, Yifan Yang, Pin Sym Foong, and Simon Perrault. 2022. Differences of Challenges of Working from Home (WFH) between Weibo and Twitter Users during COVID-19. In Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA'22). Association for Computing Machinery, New York, NY, USA, Article 259, 1–7. https://doi.org/10.1145/3491101.3519790

Acknowledgements

This thesis would not have been possible without the love and support of many individuals and groups who formed my support system. They offered guidance and support, both emotionally and professionally, playing a crucial role in my journey. In particular, I would like to express my thanks to:

1. My PhD advisor: Dr. Simon Perrault. In early 2019, while seeking PhD positions in HCI, I faced significant challenges. My skills in programming, coding, and data analysis, though rare, were underdeveloped, and my research experience was almost zero. Moreover, my spoken English was very poor, making it difficult to find a PhD position. Fortunately, I met Simon, whose encouragement played a pivotal role in my journey, fostering a sense of self-belief despite my initial doubts. Simon's mentorship style was a departure from the traditional, control-oriented approaches I had encountered in my previous experience, which often stifled creativity. He offered a balance of reliable support and appropriate autonomy in guiding my research, aligning with my personal initiative. This approach not only fostered my rapid growth but also equipped me with several key skills, to name a few:

   - Independence: Simon started to teach me to take responsibility for my research outcomes after my first several projects. That made me realize that advisors, despite their knowledge, are not infallible in every areas, and that I must exercise my own judgment. This independence is crucial for long-term success in research. A PhD may last only 4-5 years, but with this skill, I can forge a path far into the future.
   - Open-mindedness and Respect: Simon's open-mindedness, especially in acknowledging the validity of my choices even when they diverged from his advice, was enlightening. His willingness to support uncharted research topics and learn alongside me was empowering. This experience hastened my development into an independent researcher, fostering courage in my convictions and a readiness to learn from both successes and failures.
   - Networking and Equal Communication: Since joining Shuailab in Sep 2019, Simon's guidance on my first project included connecting me with Dr. Pin Sym. This collaboration was intended to equip me with his quantitative skills and Pin Sym's qualitative expertise. This experience also equipped me with the ability to discern and synthesize diverse expert opinions, a skill I will delve into later. Additionally, Simon's respectful communication fostered a friendly atmosphere, encouraging open expression of ideas.

- Non-Competitive Environment: Simon cultivated a non-competitive lab environment. As a Chinese student accustomed to intense competition and judgment, this was a refreshing change. The lab's relaxed atmosphere, devoid of hierarchy, allowed me to freely express my ideas and choose my research topics confidently.

In summary, Simon's mentorship was instrumental in shaping my PhD experience, teaching me invaluable lessons in independence, open-mindedness, effective communication, and the importance of a healthy, collaborative research environment.

2. My academic mentors: Dr. Kenny Choo, Prof. Shengdong Zhao, and Dr. Toby Jia-jun Li (in the order of when I worked with them). Each played a pivotal role in my development as a researcher, although they were not my official advisors.

- Dr. Kenny Choo was the first to guide me. He played a crucial role in refining my initial research ideas, especially in the CoAIcoder project. His passion for transforming my primary and rough ideas into actionable studies and system designs was invaluable. In the early stages of my PhD, he provided numerous materials and books, and even translated complex research concepts for me when communication using English was challenging. This support was instrumental in my first significant steps towards research. Beyond academic guidance, Kenny also offered invaluable advice during my job search and application process.
- Prof. Shengdong Zhao was the next mentor I encountered. Although he is a "big name" in HCI both in Singapore and globally, he invested substantial commitment to student mentorship. He excelled in presenting research, capable of articulating many details with clarity. His research style was thorough and detail-oriented, and the research papers under his guidance were notable for their meticulous, solid, and systematic approach. Working with him and his PhD student Nuwan, I also learned to write research papers scientifically and concisely. Additionally, being part of the vibrant HCI community he fostered in Singapore felt like coming back to a warm family.
- Finally, my interactions with Dr. Toby Jia-jun Li enriched my academic journey further. He taught me how to craft papers that are both interesting and convincing for reviewers. His extensive network in HCI and NLP opened new doors for me, including the opportunity to visit the United States and immerse myself in its academic culture. His encouragement and recognition of my achievements bolstered my confidence. His vision and taste in research have been particularly inspiring, motivating me to pursue cutting-edge research globally.

Each of these mentors has uniquely contributed to my growth and success in every stage of my PhD. Their combined influence has been a cornerstone of my development as a researcher. Lastly, I am grateful to Prof. Thomas W. Malone for offering me an excellent postdoctoral position. His research ambitions align closely with mine, instilling confidence in me to pursue my future research career with enthusiasm and determination.

3. My academic collaborators and friends: I would also like to express my sincere gratitude to my many collaborators and academic friends: Dr. Pin Sym Foong, Dr. Nuwan Janaka, Zheng Zhang, Dr. Marie Therese Siew, Dr. Katherine Fennedy, Yeo Shunyi, Gionnieve Lim, Chen Zhou, Ruyuan Wan, Yunpeng Bai, Runze Cai, Yang Chen, Lin Zhu, Yifan Yang, Shan Zhang, Danny Hyeongcheol Kim, Yuwen Lu, Zheng Ning, Simret Araya Gebreegziabher, Zhuoqun Jiang, Sharmayne, Haw Yuh, Atima Tharatipyakul, Pavithren Pakianathan, Nurhadi Ahmad and many other lab members from SHUAILab, NUS-HCI Lab and SaNDwich Lab! Their collaboration and support have been invaluable to my academic and personal growth. Some of these professional relationships have become deep personal friendships, enriching both my research journey and my life. I wish them great success in their respective study and career journeys!

4. My life partner: I am deeply thankful to my life partner, Junming Cao, for his unwavering support throughout the demanding period of my PhD studies. There were moments in the early stages of my academic journey when I struggled with unpublished papers, leading to immense frustration and self-doubt. I even was thinking of quitting my PhD to pursue an industry job. During these challenging times, Junming was my pillar of strength, constantly reminding me that publication was not the sole measure of success. He emphasized that my innate characteristics, such as curiosity and perseverance in academics, made me exceptionally suited for a research career. Furthermore, Junming reassured me that the research topics I had chosen and the work I had accomplished were valuable, even if they had yet to gain recognition in the academic community. His support went beyond emotional; he frequently provided objective feedback on my research ideas and many immature thoughts, and actively assisted me in acquiring a wide range of programming and technological skills, significantly accelerating my growth and development. His belief in my abilities and potential has been a crucial factor in my journey and success.

5. My family: I owe immense gratitude to my family for their unwavering support. My parents, Jianpeng Gao and Honglan Zhao, along with my younger brother, Jie Zhao, have been sources of my strength. My mother, Honglan, played an essential role in shaping my ambition and enthusiasm for learning from a young age. She consistently provided thoughtful consideration for my long-term development. My

father, Jianpeng, has been a constant source of unconditional love and support, fostering a nurturing environment for me to grow and pursue my dreams. My younger brother, Jie, has always been a source of pride, cheering me on at every step of my journey. I am also profoundly grateful to my grandparents, who raised me during my early years and provided a foundation of unconditional love. Their care contributed to a joyful childhood, shaping my optimistic outlook and giving me the courage and resilience to face life's challenges. Additionally, I extend my deepest thanks to my uncle, Yong Yang. His guidance and encouragement were instrumental in my decision to pursue advanced academic degrees, including my master's and PhD. His influence has been a guiding light in my academic path.

Thank you to everyone who has been a special part of my life.

Contents

List of Figures

List of Tables

List of Abbreviations

QA	**Q**ualitative **A**nalysis
CQA	**C**ollaborative **Q**ualitative **A**nalysis
AIQCs	**AI**-assisted **Q**ualitative **C**oding systems

With love and gratitude, I dedicate this thesis to
my grandparents, Chengguo Gao & Xiaomei Shi,
my parents Jianpeng Gao & Honglan Zhao,
my uncle Yong Yang,
and my life partner Junming Cao

whose commitment to my happy childhood, sources of courage and confidence, and resilience, have made it possible for me to pursue a PhD degree, and a life with curiosity and interest, and even, to change the world, in my own way.

Chapter 1

Introduction

1.1 Research Motivations

- **Human**: *"I'm trying to write a poem, but I'm stuck on the last verse. It's just not coming together."*
- **AI**: *"How about I suggest some rhymes and structures, and you can infuse them with your emotions and experiences?"*
- **Human**: *"That sounds great! Your technical help with my artistic vision - we're quite the team!"*
- **AI**: *"Absolutely! Together, we can create a masterpiece that resonates both technically and emotionally."*

The aforementioned conversation between a human and AI is an example of how such collaboration can yield intriguing outcomes, like the creation of poems, essays, and other complex works.

Human-AI collaboration has garnered significant attention in recent years. In this area, AI's role is not just to fully automate tasks traditionally performed by humans, but also to combine the ability of humans and AI to achieve tasks that cannot be finished by only AI or only humans. From a technological standpoint, consider the example of self-driving cars. While full automation could be risky in certain scenarios, a collaborative approach, where humans and autonomous vehicles work together, offers numerous benefits. In such a setup, the human operator can monitor the system and intervene in risky or complex situations, such as when the driving system loses control or faces unpredicted challenges. This collaboration not only enhances efficiency and safety but also potentially reduces human energy costs. From a societal perspective, human-AI collaboration can be instrumental in job creation, counteracting the notion that AI will replace human jobs. Instead of viewing AI as a replacement, it can be seen as a complement to human skills, where the combination of human intuition and machine efficiency can lead to superior outcomes that neither can achieve alone. Figure 1.1 illustrates a scenario in which humans and AI study together.

FIGURE 1.1: A depiction of Human-AI collaboration, in which a girl and a robot are studying together happily. This figure is generated using GPT-4 and DALL·E.

Meanwhile, data surrounds us globally, every day, playing a crucial role in various domains. It has become akin to 'fuel' that drives businesses and research towards the right direction[1]. Data science encompasses diverse types of data, including quantitative and qualitative[2] (see more in Figure 1.2). A primary objective of data science is to extract pivotal ideas and gain meaningful insights from this data. In particular, qualitative data, or unstructured data, such as interview transcripts, observation notes, and meeting notes, are particularly common in fields like education, psychology, and social sciences (Nassaji, 2015).

In today's era, where data is expanding at an unprecedented rate, understanding how to leverage AI is critical. Equally important is utilizing human expertise in data analysis to collaborate effectively with AI, thereby scaling the time-consuming and effort-intensive processes. Consequently, a new and emerging research area, AI-assisted qualitative analysis, has been developed. Researchers in this field have primarily pursued two main directions. The first involves designing tools and systems to enhance the efficiency of qualitative analysis. These tools aim to facilitate large-scale data interpretation and support the discussion process, with the primary goal of assisting users rather than automating the coding process entirely (Rietz and Maedche, 2021; Gebreegziabher et al., 2023; Hong et al., 2022). The second direction concentrates on automating the coding process by leveraging the prompting engineering and generative capabilities

[1] https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-data-types-in-statistics-
#h-introduction

[2] https://www.qualitydigest.com/inside/six-sigma-article/
types-data-and-scales-measurement-071520.html

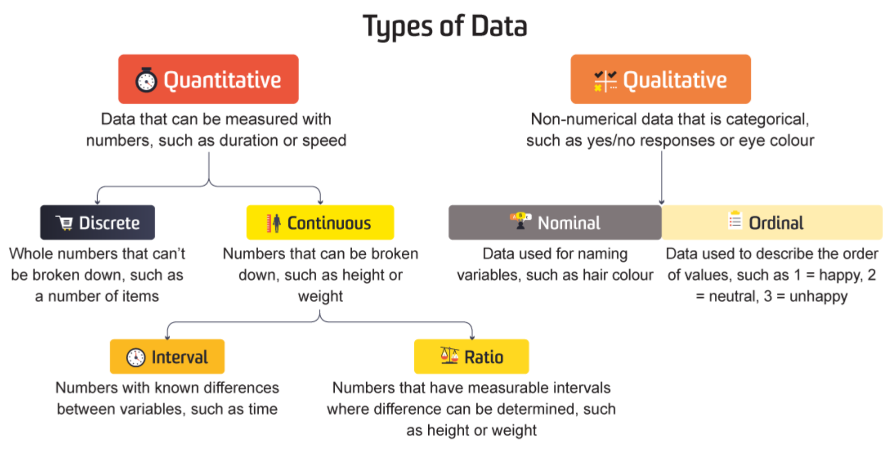

Types of Data
Quantitative
Data that can be measured with numbers, such as duration or speed
Qualitative
Non-numerical data that is categorical, such as yes/no responses or eye colour
Discrete
Whole numbers that can't be broken down, such as a number of items
Continuous
Numbers that can be broken down, such as height or weight
Nominal
Data used for naming variables, such as hair colour
Ordinal
Data used to describe the order of values, such as 1 = happy, 2 = neutral, 3 = unhappy
Interval
Numbers with known differences between variables, such as time
Ratio
Numbers that have measurable intervals where difference can be determined, such as height or weight

FIGURE 1.2: Overview of Various Data Types. The figure is from the QualityDigest website.

of Large Language Models (LLMs) (Byun, Vasicek, and Seppi, 2023; Xiao et al., 2023; Kok, 2023). Both directions are showing rapid growth and hold great potential. In this thesis, my primary focus was on the first direction: designing and developing human-AI collaborative tools for qualitative analysis.

1.2 Research Goals

In this thesis, I delve into the domain of human-AI collaboration, with a specific focus on AI-assisted qualitative analysis, grounded in foundational theories such as Grounded Theory (J. M. Corbin and Strauss, 1990) and Thematic Analysis (Maguire and Delahunt, 2017). My research objectives are twofold. Firstly, I aim to explore the potential role of AI in facilitating human-to-human collaboration within the realm of qualitative analysis. Secondly, I focus on examining the dynamics of human-AI trust and reliance, and how these factors influence the design and development of collaborative AI tools.

1.2.1 Term Definitions and Related Terms

- **Qualitative Analysis.** Qualitative Analysis involves the examination of non-numerical, unstructured data collected through methods such as observations, one-to-one interviews, and focus groups[3]. This process is guided by various theoretical frameworks, including Grounded Theory (J. M. Corbin and Strauss, 1990) and Thematic Analysis (Maguire and Delahunt, 2017). While acknowledging the inherent subjectivity in qualitative methods, these approaches emphasize the systematic and consistent interpretation

[3]https://www.questionpro.com/blog/qualitative-data/

of data, ensuring that conclusions are representative of multiple perspectives and align with other data like quantitative data (Lazar, Feng, and Hochheiser, 2017a).

- **Collaborative Qualitative Analysis (CQA).** CQA is a process in which there is joint focus and dialogue among two or more researchers regarding a shared body of data, to produce an agreed interpretation (Gao, Choo, et al., 2023; Cornish, Gillespie, and Zittoun, 2013).
- **Code:** A code is typically a succinct word or phrase created by the researcher to encapsulate and interpret a segment of qualitative data. This facilitates subsequent pattern detection, categorization, and theory building for analytical purposes (Saldaña, 2021).
- **Coding**. Qualitative Coding serves as a key method for analyzing qualitative data. It involves labeling segments of data with codes that concurrently categorize, encapsulate, and interpret each individual data point (Flick, 2013).
- **Coders**. Researchers engaged in the process of qualitative coding are called coders. A coding team usually includes two to three coders.
- **Codebook/Themes/Code groups:** A codebook is a hierarchical collection of code categories or thematic structure, typically featuring first and second-order themes or code groups, definitions, transcript quotations, and criteria for including or excluding quotations (Richards and Hemphill, 2018; Saldaña, 2021).
- **Agreement/Consensus:** Agreement or consensus is attained through in-depth discussions among researchers, where divergent viewpoints are scrutinized and potentially reconciled following separate rounds of dialogue (N. McDonald, Schoenebeck, and Forte, 2019). The degree of agreement among multiple coders serves as an indicator of the analytical rigor of a study (Cornish, Gillespie, and Zittoun, 2013).
- **Intercoder Reliability (IRR):** IRR is a numerical metric aimed at quantifying agreement among multiple researchers involved in coding qualitative data (N. McDonald, Schoenebeck, and Forte, 2019; O'Connor and Joffe, 2020).
- **Independence:** Typically, open coding and initial code development are undertaken independently by individual team members to minimize external influence on their initial coding choices (Hall et al., 2005; Cornish, Gillespie, and Zittoun, 2013).
- **Data units/Unit-of-analysis:** The unit-of-analysis (UoA) specifies the granularity at which text annotations are made, such as at the flexible or sentence level (Rietz and Maedche, 2021).

- **Coding granularity**. Coding Granularity, though not universally applied, is crucial in our context. It means the variability in labels and text inherent in human coding. Specifically, *Text Granularity* refers to the chosen length and depth of text designated for coding. In contrast, *Code Granularity* means the chosen length and depth of each code. This distinction highlights the nuanced approach required in qualitative coding, addressing both the selection of text and the detailed nature of the coding itself.

- **Grounded Theory and Thematic Analysis**. Two foundational theories form the basis of our approach to qualitative analysis, offering specific steps and guidelines for effective implementation. A detailed description of these theories and their respective steps and practices will be provided in Section 3.1.2 of Chapter 3.

- **AI-assisted qualitative coding systems (AIQCs).** Researchers have sought to alleviate this labor-intensive task by harnessing Artificial Intelligence (AI), leading to the development of the AI-assisted Qualitative Coding system (N.-C. Chen et al., 2018; Marathe and Toyama, 2018b).

- **Semi-structured Interview.** A pivotal method in qualitative data collection, the semi-structured interview provides researchers with a framework to gather consistent data across participants by asking a set of predetermined questions. Simultaneously, it offers the flexibility to explore open-ended questions, allowing for deeper insights and critical commentary (Lazar, Feng, and Hochheiser, 2017b). This approach balances the need for structured inquiry with the opportunity to delve into the unique perspectives and experiences of each participant.

1.2.2 Research Questions and Scopes

The thesis is structured into three main chapters (Chapter 4, Chapter 5 and Chapter 6) with a question-finding Chapter 2, as illustrated in Figure 1.3.

Examining the Effectiveness of AI-assisted Human-to-Human Collaboration in Qualitative Analysis

In this first work, we began with a fundamental research question: In the context of collaborative qualitative analysis where AI has not been traditionally employed, is it feasible to integrate AI into this process? If so, how might AI be utilized to enhance teamwork within qualitative analysis? Furthermore, could AI act as an agent to support human-to-human collaboration? To address these questions, we have structured our investigation into three parts:

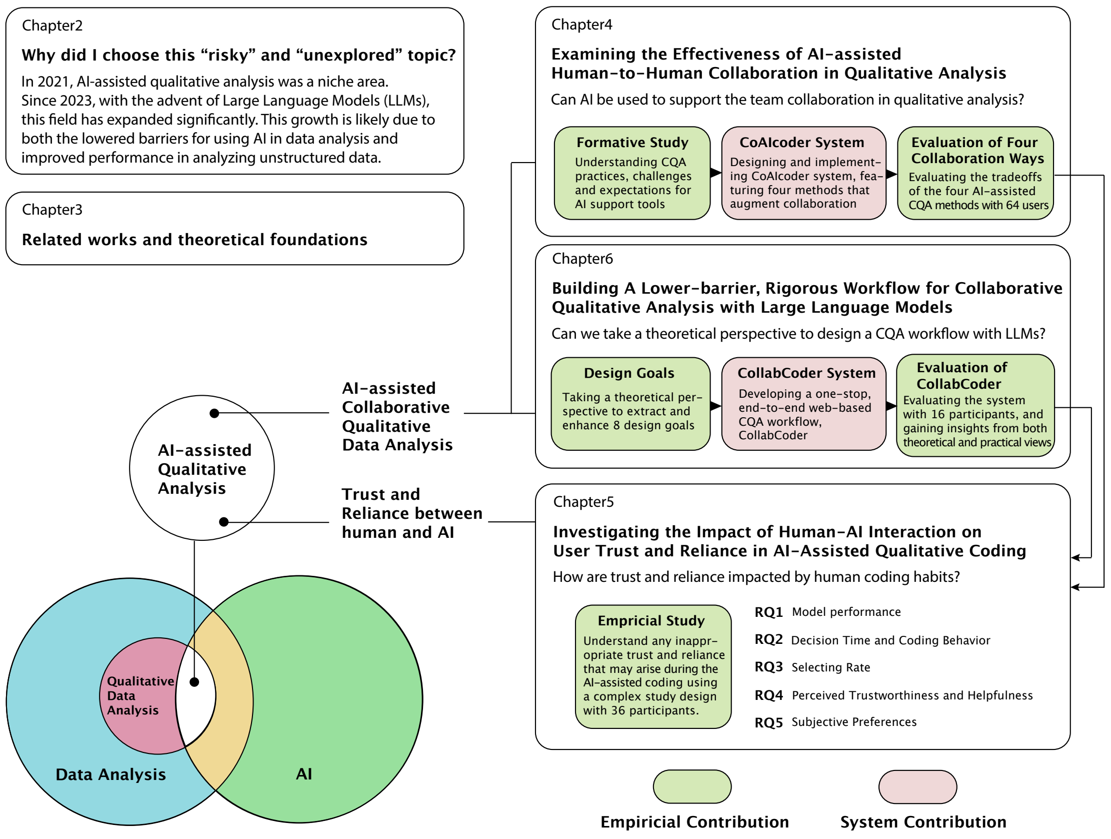

Chapter2
Why did I choose this “risky” and “unexplored” topic?
In 2021, AI-assisted qualitative analysis was a niche area. Since 2023, with the advent of Large Language Models (LLMs), this field has expanded significantly. This growth is likely due to both the lowered barriers for using AI in data analysis and improved performance in analyzing unstructured data.
Chapter3
Related works and theoretical foundations
Chapter4
Examining the Effectiveness of AI-assisted Human-to-Human Collaboration in Qualitative Analysis
Can AI be used to support the team collaboration in qualitative analysis?
Formative Study
Understanding CQA practices, challenges and expectations for AI support tools
CoAIcoder System
Designing and implementing CoAIcoder system, featuring four methods that augment collaboration
Evaluation of Four Collaboration Ways
Evaluating the tradeoffs of the four AI-assisted CQA methods with 64 users
Chapter6
Building A Lower-barrier, Rigorous Workflow for Collaborative Qualitative Analysis with Large Language Models
Can we take a theoretical perspective to design a CQA workflow with LLMs?
Design Goals
Taking a theoretical perspective to extract and enhance 8 design goals
CollabCoder System
Developing a one-stop, end-to-end web-based CQA workflow, CollabCoder
Evaluation of CollabCoder
Evaluating the system with 16 participants, and gaining insights from both theoretical and practical views
Chapter5
Investigating the Impact of Human-AI Interaction on User Trust and Reliance in AI-Assisted Qualitative Coding
How are trust and reliance impacted by human coding habits?
Empricial Study
Understand any inappropriate trust and reliance that may arise during the AI-assisted coding using a complex study design with 36 participants.
RQ1 Model performance
RQ2 Decision Time and Coding Behavior
RQ3 Selecting Rate
RQ4 Perceived Trustworthiness and Helpfulness
RQ5 Subjective Preferences
AI-assisted Collaborative Qualitative Data Analysis
Trust and Reliance between human and AI
AI-assisted Qualitative Analysis
Qualitative Data Analysis
Data Analysis
AI
Empiricial Contribution
System Contribution

FIGURE 1.3: Overview of the Thesis Content.

- A formative study aimed at understanding current behaviors and challenges in CQA, as well as gathering insights into expectations for AI support within team-based settings.
- Building upon the initial findings, we developed CoAIcoder, a tool that introduces four innovative methods designed to augment human-to-human collaboration within CQA. We also involves detailing specific design features and guidelines that could inform the development of future AI-assisted CQA systems.
- An evaluation of CoAIcoder, assessing whether our proposed methods are practical and effective across various metrics like IRR, Coding time, Code Diversity, and Code Coverage, and examining any associated trade-offs.

Investigating the Impact of Human-AI Interaction on User Trust and Reliance in AI-Assisted Qualitative Coding

In this subsequent study, we build upon the foundations laid by our initial project, CoAIcoder, to delve deeper into the nuances of user trust and reliance in AI systems within the context of qualitative analysis. The primary aim is to harmonize the AI system's functionalities with human coding practices, thereby ensuring a more intuitive and collaborative interaction. A critical aspect of this investigation is to identify and understand any instances of inappropriate reliance that may arise during the AI-assisted coding process. To advance this goal, we utilized the system developed in our previous work and formulated the following research questions for AI-assisted qualitative coding systems (AIQCs):

- RQ1. How does coding granularity impact the model performance of AIQCs?
- RQ2. How does coding granularity impact users' Decision Time and Coding Behavior when using AIQCs?
- RQ3. How does coding granularity impact users' Selecting Rate?
- RQ4. How does coding granularity impact users' Perceived Trustworthiness and Helpfulness of AIQCs?
- RQ5. How does coding granularity impact users' Subjective Preferences when using AIQCs?

Building A Lower-barrier, Rigorous Workflow for Collaborative Qualitative Analysis with Large Language Models

Rigor and in-depth interpretation are the two main goals of qualitative analysis. In this study, we engage in an extensive exploration of its theoretical underpinnings, with a special focus on methodologies such as Grounded Theory and Thematic Analysis. Our key objective is to explore whether it's feasible to develop a comprehensive, end-to-end workflow that not only integrates these theoretical approaches to ensure the rigor of CQA but also reduces the barriers to efficient collaboration. To this end, we have formulated specific design goals and considerations, leading to the creation of the CollabCoder web-based workflow, which prompts us to pose the following research questions:

- RQ1. Can CollabCoder 's workflow support qualitative coders to conduct CQA effectively?
- RQ2. How does CollabCoder compare to currently available tools like Atlas.ti Web?
- RQ3. How can the design of the CollabCoder workflow be improved?

1.3 Research Contributions

1.3.1 Systems and Tools Contributions

CoAIcoder: A Tool for Human-to-Human Collaboration via AI within Qualitative Coding Team

In Section 4.3 of Chapter 4, we present the design and development of CoAIcoder, a system that facilitates collaborative qualitative analysis between two coders. Unlike previous tools focused solely on human-AI teams, CoAIcoder enables two coders to utilize their shared coding history for synchronous or asynchronous qualitative coding. The integration of an AI agent within this system aids in establishing a shared mental model or knowledge space, enhancing the coding process.

The development and application of this system are detailed in our publication on TOCHI2023 (Gao, Choo, et al., 2023).

CollabCoder: A Tool for Collaborative Qualitative Analysis with Large Language Models

In Section 6.2 of Chapter 6, we adopt a theory-driven approach to design and develop CollabCoder, a system that enables two coders to engage in collaborative qualitative analysis. This system integrates established qualitative analysis theories into its workflow. The workflow comprises: **Independent Open Coding**: This phase involves on-demand code suggestions from LLMs, including initial code proposals and GPT-3.5-generated suggestions. **Iterative Discussion**: Focused on conflict mediation within the coding team, this phase results in a consensus on code decisions. **Codebook Development**:

Here, code groups may be formed, utilizing LLM-generated suggestions based on the decided codes.

This innovative approach has been showcased at the CSCW2023 demo track (Gao, Guo, T. J.-J. Li, et al., 2023) and is also currently under revision, targeting acceptance at CHI2024.

1.3.2 Empirical Contributions

Exploration in CQA Practices, Challenges, and Expectations

In Section 4.2 of Chapter 4, we conducted an exploratory study by interviewing 8 participants with diverse levels of experience in CQA. We then employed a thematic analysis method to analyze the interview data, which enabled us to 1) enhance our understanding of CQA behaviors and challenges; and 2) identify the potential of AI in facilitating human-to-human collaboration, which subsequently inspired the development of CoAIcoder.

These findings are published on TOCHI2023 (Gao, Choo, et al., 2023).

Evaluation of CoAIcoder with Three Factors: With/Without AI Model, Synchrony, and Shared/Not Shared Model

In Section 4.4 of Chapter 4, we conducted an evaluation of CoAIcoder involving 64 participants. The study was structured around three factors—AI model presence, synchrony in collaboration, and whether the model was shared or not—resulting in eight distinct conditions (2^3 combinations). We selected four feasible conditions among them and subsequently conducted a between-subjects design study with 64 participants to evaluate the CoAIcoder system.

Key findings from this evaluation include:

- The identification of a critical trade-off between coding efficiency and coding quality, influenced by the level of independence among coders. Specifically, lower independence (more communication among coders) tends to yield higher efficiency and initial intercoder reliability (IRR) but results in lower code diversity. Conversely, higher independence (less communication among coders) leads to lower efficiency and initial IRR but promotes greater code diversity.

- The recognition of context as a vital aspect in AI-assisted CQA. This involves understanding whether a situation prioritizes efficiency or code diversity, and questioning if maximal coding efficiency is always the desired outcome.

These findings are published at TOCHI2023 (Gao, Choo, et al., 2023).

Evaluation of User Trust and Reliance on the AIcoder System

AIcoder, a non-collaborative version of CoAIcoder, is evaluated in Chapter 5. Our investigation initially identified a mismatch between users' qualitative coding habits and the system's behavior. To delve into this discrepancy, we analyzed user interactions with our initial AI model, particularly focusing on the role of coding granularity. Consequently, we categorized coding granularity into two distinct levels and conducted a quantitative study with 36 participants. This study aimed to assess the impact of users' coding strategies on the AI model's performance and, subsequently, on users' trust and reliance on the system. Our findings reveal that:

- Qualitative coding should be viewed not as a monolithic task but as an ensemble of subtasks with varying degrees of complexity. Some subtasks (e.g., Paragraph, Long Codes) presented more challenges, while others (e.g., Short Codes, Mixed Codes, Sentence, Selective) were relatively simpler.
- A notable divergence emerged between participants' perceived helpfulness and actual reliance on the AI system. For more complex tasks, participants reported higher perceived helpfulness but demonstrated lower reliance. Conversely, for simpler tasks, they showed higher reliance but perceived them as less helpful.
- The study also illuminated the risks associated with both under-reliance and over-reliance on AIQCs. Under-reliance may prevent users from leveraging the full benefits of AIQCs, whereas over-reliance could result in narrowly focused yet superficial outcomes.

These findings are in the process of submission and publication.

Evaluation of CollabCoder

In Section 6.5, we present our comprehensive evaluation of the CollabCoder workflow, designed for supporting collaborative qualitative analysis. This evaluation, conducted through a within-subject design involving 16 participants, yielded several key findings:

- User Experience: CollabCoder was highly rated for its user-friendliness, especially beneficial for those new to the CQA process. Over 75% of participants agreed or strongly agreed that the tool is 'easy to use' and can be 'learned quickly'. The system was particularly effective in supporting coding independence, clarifying (dis)agreements, and fostering shared understanding within teams. More than 75% of participants affirmed CollabCoder's utility in identifying and resolving disagreements, understanding others' perspectives, and gauging the consensus level.
- Efficiency in Collaboration: Compared to tools like Atlas.ti Web, CollabCoder optimized the discussion phase by enabling code pairs to be resolved in a single dialogue

session. This feature significantly reduces the necessity for multiple rounds of discussion, thus enhancing collaborative efficiency.

- Role of GPT in CQA: Our findings underscore the importance of balancing LLM capabilities with user autonomy. GPT, in its role as a ′suggestion provider′ during the initial coding phase and as a ′mediator′ and ′facilitator′ during discussions, contributes to efficient and equitable decision-making and code group formation.

These findings are currently under revision, targeting acceptance at CHI2024.

Chapter 2

Choice of Research Topic

In this Chapter, I want to document my process of choosing a research topic, as suggested by Simon. This documentation is intended to serve as a reference for early-stage PhD students or researchers who are undergoing the selection of an appropriate topic. This is particularly relevant when the chosen topic diverges slightly from the advisor's past expertise and lies in a niche, albeit promising, area of study.

2.1 My Research Journey during the COVID-19 Outbreak

In early 2020, just before the Chinese New Year, I returned to my serene hometown in Yuncheng, Shanxi Province, China, to celebrate with my family. However, the sudden escalation of COVID-19 into a global pandemic disrupted everything. At that point, I had not started my PhD research and was uncertain about my direction. My aspiration was to conduct groundbreaking, impactful research that could change the world.

Yet, my advisor proposed a project examining the challenges of working from home in China during the pandemic. Initially, I was hesitant; my interest in Human-Computer Interaction leaned more towards mobile sensing, wearable technology, and innovative user support systems, as reflected in my master's work on Expressive Plant (Gao, Zhou, et al., 2018). This new project seemed distant from my envisioned path, but with travel restrictions and a national lockdown preventing my return to Singapore, it appeared to be the most viable option for continuing my research.

Seeking guidance, I inquired in a class about how PhD students select their research areas. A response from Prof. Shaowei Lin[1] in my Statistics class resonated with me. He suggested that research interests often evolve significantly over time. This insight helped me realize the importance of engaging in hands-on research. While selecting significant topics for research is essential, it is important to recognize that your interests may evolve over time. What remains consistent is the experience gained from active participation in the research process, which encompasses everything from experimental

[1]https://shaoweilin.github.io/

design to writing a comprehensive paper. This understanding marked a turning point in my academic journey.

2.2 Two Questions to Qualitative Analysis

When I started my first research project, it involved conducting interviews, understanding participant needs, performing qualitative analysis, and compiling a paper. This method differed from my technological and design-oriented background. Holding a Bachelor's and a Master of Engineering degrees, my background primarily lies in these areas. Yet, the project's focus on qualitative analysis and interviews was more akin to social science methodologies, marking a significant shift from my usual field of work.

During this time, I was advised by Dr. Pin Sym Foong[2], an HCI researcher from the National University of Singapore with much experience in qualitative analysis. This experience made me understand the differences in perspectives among people from varied disciplines. My thought process, shaped by my technological background, became more objective. In technology, solutions are more definitive and less subject to the varied perspectives that are common in disciplines like social sciences; for example, designing a circuit diagram requires specific configurations to function.

Therefore, **qualitative analysis**, a widely utilized method for analyzing interview data, revealed two aspects that particularly surprised me:

- **Subjectivity in Interpretation**: Qualitative analysis is inherently subjective. I observed that individuals with different backgrounds can have varying understandings or interpretations of the same data. This diversity in perspective was starkly different from the more objective nature of technological solutions.
- **Time Consumption for Analysis**: The process requires a significant time investment, often spanning months, to analyze relatively small datasets. This seemed particularly inefficient and unreasonable in an era where machines and AI are capable of automating numerous tasks. This contrast between manual analysis and technological efficiency was particularly striking.

I found myself increasingly drawn to the above two intriguing questions that had emerged from my initial foray into qualitative analysis. But at that time, I wasn't sure who could address these concerns.

After completing the first paper on the challenges of working from home (WFH) during COVID-19, I submitted it to CHI2021[3], but unfortunately, it was rejected. At that point in my journey, I had completed courses in Statistics, Machine Learning, and

[2]https://pinsym.wordpress.com/
[3]https://chi2021.acm.org/

Research Methods, but lacked practical experience in AI. Seeking deeper understanding, I reached out to several professors, including Prof. Lu Wei[4] and Prof. Kwan Hui Lim[5]. Through these interactions, I learned about BERT and the advancements in NLP technology. Motivated by this newfound knowledge, I attempted to apply NLP techniques, such as LDA modeling, to build a topic model for a social media dataset. This effort is part of an experiment in my WFH-related paper, which was initially rejected but is still an ongoing project. With my advisor's support, I drafted an extensive paper and submitted it to IJHCS, but got rejected again.

This setback was particularly disheartening as I was entering the third year of my PhD without any significant publications in top-tier journals. This experience prompted me to introspect and critically evaluate my research methodologies. It raised important questions about the true promise of this direction: Is it truly promising and suitable for me, or does it merely appear promising without truly being so? Alternatively, could it be promising but not the right fit for my skills and interests? Eventually, I decided to shift my focus away from this research direction. To sum up this project, I condensed my findings into a short paper format and published it on the CHI2022 late-breaking work track.

2.3 A New Start: Using AI for Qualitative Analysis

During this period of frustration and self-doubt, the two pressing questions resurfaced in my mind: Why not address the efficiency issue in qualitative analysis, a critical method in HCI, using cutting-edge technology like BERT? As qualitative analysis is a fundamental method in HCI and numerous other fields, this issue is of significant importance and warrants the investment of my efforts to make a potentially impactful contribution. Although my previous attempt at integrating technology with qualitative analysis didn't yield big success, I realized it was time to leverage my strengths. My undergraduate and master's studies had honed my skills in tool-building, and I was confident in my ability to rapidly learn new technologies. Even if I wasn't an expert, I knew I could collaborate with others to overcome technical challenges.

This renewed perspective coincided with a lab meeting where I encountered Dr. Kenny[6] discussing the intersection of HCI and NLP in research. It resonated deeply with my thoughts. Consequently, I engaged in several conversations with Kenny, who then advised me to start attending weekly meetings with a group of professors, including Simon, Prof. Roy Lee[7], and himself. I seized this opportunity and, during

[4]https://istd.sutd.edu.sg/people/faculty/lu-wei
[5]https://people.sutd.edu.sg/~kwanhui_lim/
[6]https://kennychoo.net/
[7]https://info.roylee.sg/

these meetings, shared insights and papers on Human-AI interaction and collaboration. At that time, these topics were still emerging in the field, offering a fresh and exciting research avenue.

2.4 A Paper on This Topic Emerges at CHI

In early 2021, a CHI paper by Rietz et al. (Rietz and Maedche, 2021) left me feeling disheartened, as it appeared that my idea had already been explored. This led to a period of reflection and exploration spanning several weeks, even the entire summer, as I pondered my next steps. The application of AI and classification models, like SGD techniques, in providing code suggestions for text data, was already a reality. I needed to identify my unique research gap.

When my attempts to obtain code from the authors of the CHI paper yielded no response, I did not give up and decided to develop a similar tool using a different approach, focusing on intent recognition. This choice was strategic, as there are numerous existing APIs for intent recognition provided by major companies like Facebook and Google, which promised to simplify the development process.

2.5 At Last: Pathway to Integrating AI in Qualitative Analysis

Following a few weeks of exploration, I presented my preliminary findings and the tools I had investigated to Kenny. In response, he proposed an interesting angle: exploring the collaborative aspect of the tool, specifically how it could enable two researchers to work together on coding tasks. Kenny was quite enthusiastic about this idea, but to me, it seemed too simplistic and lacked the innovative edge I was seeking. It felt like an "easy" research path, and I was aiming for something more challenging. Consequently, I declined Kenny's suggestion.

I then approached Simon for advice on the viability of this research direction. He did not give me a concrete answer at that time, which I understood. Predicting the success of a new research path is difficult, and there are no guarantees of success, especially in uncharted territories.

However, I soon reconsidered my stance. Although the idea appeared simple, it was undoubtedly feasible. Even if the innovation wasn't groundbreaking, I didn't have any other significant ideas at hand. So, I thought, why not give it a try? Perhaps by delving into it, I might uncover something truly intriguing. I decided to pursue this path with the mindset that if a better idea emerged, I could always pivot towards it.

At that time, there was a risk that this new direction might also not yield the desired results, just like my previous endeavors. But fortunately, this was not the case. The field started to gain traction and became increasingly popular. Being among the first to venture into this exciting, new, and emerging area of research felt exhilarating.

Now, let me begin to delve into the details of my PhD work.

Chapter 3

Background and Related Work

3.1 Qualitative Analysis and Its Methods

3.1.1 What is Qualitative Analysis?

Qualitative research is widely embraced across various disciplines, including but not limited to social science, anthropology, political science, psychology, educational research, and human-computer interaction (Charmaz, 2014; J. Corbin and Strauss, 2014; Lazar, Feng, and Hochheiser, 2017a). It is an important methodology for interpreting data from interviews, focus groups, observations, and more (Lazar, Feng, and Hochheiser, 2017b; Flick, 2013). The goal of qualitative analysis is to transform unstructured data into detailed insights regarding key aspects of a given situation or phenomenon, addressing researchers' concerns (Lazar, Feng, and Hochheiser, 2017b). Commonly employed strategies include Grounded Theory (Flick, 2013) and thematic analysis (Maguire and Delahunt, 2017).

3.1.2 Methodology: Grounded Theory, thematic analysis, and others

Grounded Theory (GT), originally formulated by Glaser and Strauss (Glaser and Strauss, 2017; Flick, 2013). Its primary objective is to abstract *theoretical conceptions* based on *descriptive* data (J. Corbin and Strauss, 2008; Bryant and Charmaz, 2007). A primary approach in GT involves *coding*, specifically assigning *codes* to data segments. These conceptual *codes* act as crucial bridges between descriptive data and theoretical constructs (Bryant and Charmaz, 2007). In particular, GT coding involves two key phases: initial and focused coding. In initial coding, researchers scrutinize data fragments—words, lines, or incidents—and add codes to them. During focused coding, researchers refine initial codes by testing them against a larger dataset. Throughout, they continuously compare data with both other data and existing codes (Charmaz, 2014), in order to build theoretical conceptions or theories. Similarly, thematic analysis is another method commonly used for analyzing qualitative data, aimed at identifying, analyzing, and elucidating recurring themes within the dataset (Braun and Clarke, 2006; Maguire and Delahunt, 2017).

Charmaz (Charmaz, 2014) presented two phases in the Grounded Theory: *Initial Coding* (or *Open Coding*) and *Focused Coding*. Serving as the preliminary step in transitioning from raw data's concrete ideas and concepts to formulating analytic interpretations (Charmaz, 2014), *Open Coding* involves assigning a summarizing label to varied segments of data, with sizes ranging from a single word to a full paragraph (Charmaz, 2014; Saldaña, 2021; J. Corbin and Strauss, 2014; DeCuir-Gunby, Marshall, and McCulloch, 2011). Subsequently, these labels or codes undergo thorough discussion within a team, leading to the development of a codebook. This codebook comprises a variety of labels/codes correlating with the raw data, thereby facilitating further data analysis (DeCuir-Gunby, Marshall, and McCulloch, 2011).

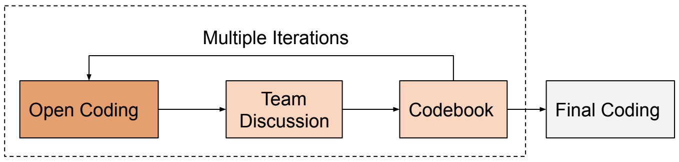


FIGURE 3.1: A Circular Coding Process (See More in DeCuir-Gunby, Marshall, and McCulloch, 2011; Richards and Hemphill, 2018; Saldaña, 2021).

Nonetheless, the coding process is labor-intensive and demands significant effort, often necessitating multiple iterations within a team (Marathe and Toyama, 2018b; Rietz and Maedche, 2021; Yan, McCracken, and Crowston, 2014). This is because the development of the codebook is not a linear process but rather cyclical in nature (see Figure 3.1), during which *Open Coding* acts as a key step (Charmaz, 2014; DeCuir-Gunby, Marshall, and McCulloch, 2011) and is inevitably revisited multiple times. Often, researchers need to revisit raw data, potentially collecting more data and conducting additional coding until reaching saturation, ensuring no nuanced information is overlooked.

The aforementioned open coding resembles an inductive approach, where patterns in the data are identified during the coding and labeling process. In contrast, focused coding aligns more with a deductive approach, involving coding based on a pre-existing theory (Azungah, 2018; Fereday and Muir-Cochrane, 2006). A detailed description of the two approaches is presented in Figure 3.2.

3.1.3 Collaborative Qualitative Analysis

CQA plays an important role in qualitative research and fosters robust and dependable interpretations of qualitative data (Anderson, Guerreiro, and J. Smith, 2016; Richards and Hemphill, 2018; Flick, 2013). Cornish et al. provide a more detailed definition of

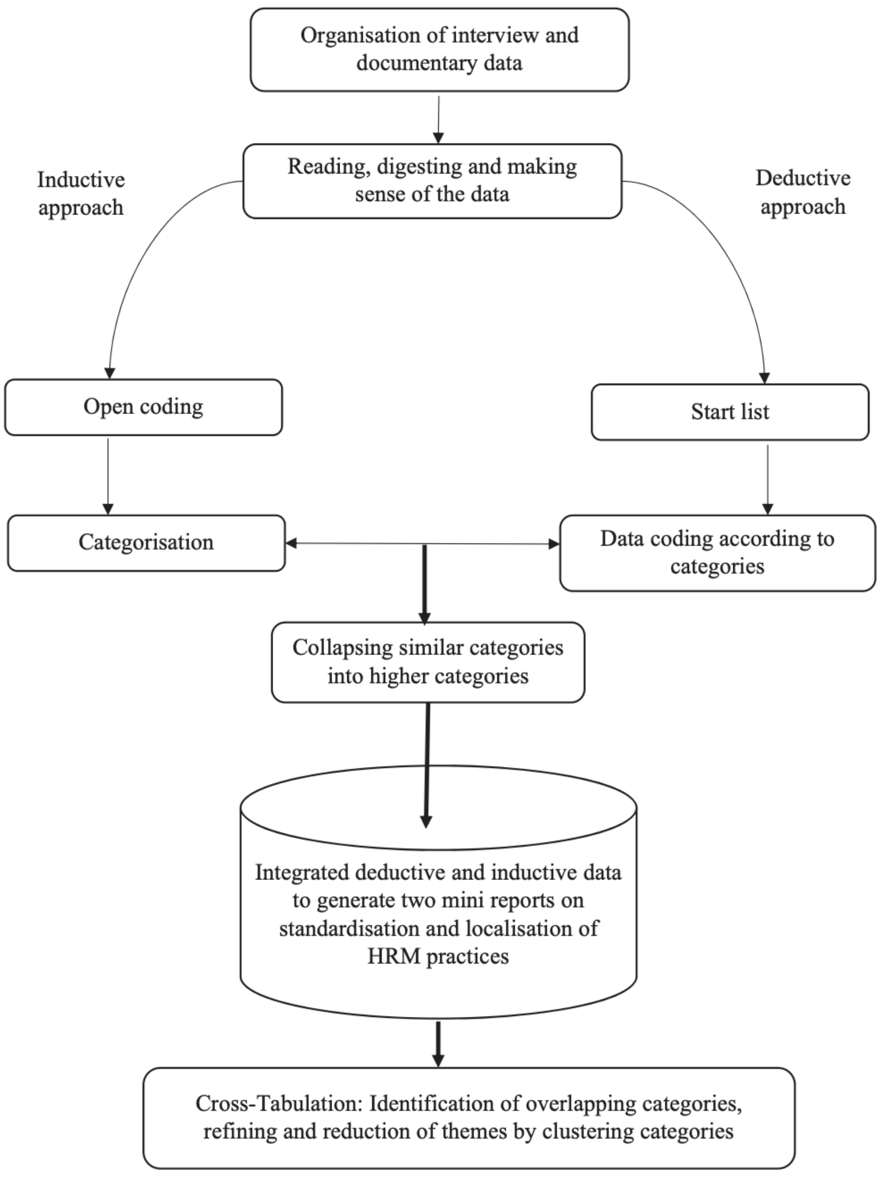

Organisation of interview and documentary data
Reading, digesting and making sense of the data
Inductive approach
Deductive approach
Open coding
Start list
Categorisation
Data coding according to categories
Collapsing similar categories into higher categories
Integrated deductive and inductive data to generate two mini reports on standardisation and localisation of HRM practices
Cross-Tabulation: Identification of overlapping categories, refining and reduction of themes by clustering categories

FIGURE 3.2: Preparation, organisation and qualitative data analysis process. Figure from Azungah, 2018.

CQA, describing it as *"a process in which there is joint focus and dialogue among two or more researchers regarding a shared body of data, to produce an agreed interpretation"* (Cornish, Gillespie, and Zittoun, 2013).

Several practical frameworks exist for conducting CQA (Hall et al., 2005; Bryant and Charmaz, 2007; Richards and Hemphill, 2018). Particularly, Richards et al. (Richards and Hemphill, 2018) have proposed a six-step methodology rooted in GT and thematic analysis. The methodology encompasses the following steps: ① preliminary organization and planning: An initial team meeting outlines project logistics and sets the overall analysis plan; ② open and axial coding: Team members use open coding to identify concepts and patterns, followed by axial coding to link these patterns (J. Corbin and Strauss, 2008; Flick, 2013); ③ development of a preliminary codebook: One team member reviews the memos and formulates an initial codebook; ④ pilot testing the codebook: After creating the initial codebook, it is tested on new data. Researchers independently code 2-3 transcripts and note codebook issues in their journals; ⑤ final coding process: The updated codebook is applied to all data, including initially-coded transcripts; and ⑥ review and finalization of the codebook and themes: After coding all transcripts, either by consensus or split coding, the team holds a final meeting to finalize the codebook.

Richards et al. also delineate two distinct CQA approaches: *consensus coding* and *split coding*. *Consensus coding* is more rigorous but time-consuming; each coder independently codes the same data and then engages in a team discussion to resolve disagreements and reach a consensus. Conversely, *split coding* is quicker but less rigorous, with coders working on separate data sets. This method leans heavily on the clarity established during the preliminary coding phases and pre-defined coding conventions.

3.2 Human, Traditional AI, and Qualitative Analysis

3.2.1 Definition of Human-AI Collaboration, Human-AI Interaction, and Human-Centric AI

We are witnessing an era where AI, once a broadly accessible technology, is increasingly becoming the domain of large corporations due to the high costs associated with training large models. Concurrently, there's a shift towards human-centered AI. Many big companies and universities are establishing their own human-centered AI guidelines and departments. As an emerging field, human-AI interaction, human-AI collaboration and human-centered AI have gained significant attention following the rise of models like GPT. These terms frequently headline blogs, articles, and news stories. However, to the best of my knowledge, no universally accepted definition differentiates these terms or the nuances they represent.

A term that seems to encapsulate these concepts, as proposed in Ben Shneiderman's book (Shneiderman, 2022), has gained wide acceptance. This term, Human-Centered AI (Artificial Intelligence), is defined as *'focusing on enhancing human performance, making systems reliable, safe, and trustworthy.'* This approach contrasts with traditional AI, which primarily focuses on automating tasks traditionally performed by humans, such as pattern recognition, language processing and translation, speech and image generation, and strategic games like chess (Shneiderman, 2020).

If the essence of human-centered AI lies in enhancing AI's reliability, safety, and trustworthiness, then human-AI collaboration is about improving the collective performance of teams comprising both humans and AI. Wang et al.'s definition in CHI2020 (Wang et al., 2020) adapts the concept of traditional collaboration—emphasizing mutual goal understanding, task co-management, and shared progress tracking, typically among humans—to the context of human-AI collaboration. By adopting a Computer-Supported Cooperative Work (CSCW) perspective (Wang et al., 2020), integrating AI into human workflows can potentially exceed the capabilities of humans or AI working solo (Campero et al., 2022).

However, the Interaction Design Foundation[1] offers a slightly different perspective, defining Human-AI Interaction as the study and design of communication and collaboration between humans and AI systems. This definition appears to overlap considerably with that of human-AI collaboration. The distinguishing factor, however, may lie in the emphasis of each. Human-AI interaction, as the name suggests, might focus more on designing interactive technologies that facilitate collaboration between humans and AI. This approach aligns closely with the ethos of UIST (ACM Symposium on User Interface Software and Technology)[2], which concentrates on innovations in human-computer interfaces.

3.2.2 (Semi)-Automating Qualitative Analysis

For decades, there has been a substantial effort to integrate AI into qualitative analysis, signifying a long-standing history of such endeavors. Before delving into our review of the current landscape of LLMs in qualitative analysis, it's crucial to acknowledge some of the initial attempts in this field. While these early technologies might not have achieved the level of performance seen in models like GPT-4, understanding them is essential for building a foundational knowledge of the evolution of this research area.

One main category is to use (semi)automated techniques to facilitate qualitative analysis. Some of these studies (Marathe and Toyama, 2018b; Crowston, X. Liu, and

[1] https://www.interaction-design.org/literature/topics/human-ai-interaction#:~:text=Human%2DAI%20interaction%20can%20ensure,promote%20mutual%20understanding%20and%20cooperation.

[2] https://uist.acm.org/2023/

E. E. Allen, 2010; Paredes et al., 2017) suggest utilizing code rules for extracting pertinent sections from a given text. For instance, a Boolean rule for `Definition of arts` may be constructed by linking various keywords using Boolean operators such as AND, OR, and NOT (e.g., `(definition OR define OR constitute) AND art`) (Marathe and Toyama, 2018b). This rule is then evaluated against a target text, and a match is identified if their similarity surpasses a specified threshold.

Moreover, scholars propose code pattern auto-detection in order to support more flexibility (Nelson, 2020; Gebreegziabher et al., 2023). For instance, Nelson's three-step method (Nelson, 2020) applies unsupervised machine learning for data pattern discovery, enhancing scalable, exploratory analysis. Meanwhile, PaTAT, introduced by Gebreegziabher et al. (Gebreegziabher et al., 2023), finds user coding patterns in real-time, predicting future codes. These studies indicate that auto-detection of code rules for partial automation shows significant potential.

Furthermore, unsupervised machine learning approaches, such as topic modeling (Baumer et al., 2017; Leeson et al., 2019; Felix, Dasgupta, and Bertini, 2018; Hong et al., 2022), have proven valuable for detecting topics or labels in qualitative data, especially when dealing with large-scale datasets. By identifying statistical regularities within text, topic modeling can discern thematic patterns, yielding results akin to traditional grounded theory methods (Leeson et al., 2019; Muller et al., 2016). This approach allows researchers to uncover topics or labels during the early stages of qualitative analysis more effectively (Felix, Dasgupta, and Bertini, 2018; Nguyen et al., 2021; Kaufmann, Barcomb, and Riehle, 2020).

In addition, supervised techniques such as text classification has gained widespread usage in qualitative analysis. For instance, Yan et al. (Yan, McCracken, and Crowston, 2014) utilized Support Vector Machine (SVM) classification, using pre-selected features and parameters. They trained the SVM model with codes provided by human coders to classify large-scale text data. In a similar vein, Rietz et al.'s Cody (Rietz and Maedche, 2021) used a logistic regression model, employing stochastic gradient descent (SGD) learning. This model was trained to categorize unseen data based on existing annotations.

3.2.3 (Semi)-Automating Collaborative Qualitative Analysis

While these initial methods and efforts in AI for qualitative analysis are noteworthy, progress in this area has been gradual, primarily due to limitations in AI performance. This bottleneck has led to a lack of confidence in developing truly effective tools. Consequently, many studies have concentrated on specific key aspects of qualitative analysis like collaboration (Gao, Choo, et al., 2023).

For example, Zade et al. (Zade et al., 2018) proposed a strategy that enables coders to order the degrees of consensus, using tree-based ranking metrics to quantify coding ambiguity. This ordering can extend from the most ambiguous to the least, or from low to high agreement. Likewise, *Aeonium*, introduced by Drouhard et al. (Drouhard et al., 2017), is a visual analytics system that assists team members with tools to review, edit, and add code definitions and examples shared within the team. It also enables monitoring of the coding history throughout the process, with the aim of revealing disagreements and reducing ambiguity. Furthermore, Ganji et al. (Ganji, Orand, and D. W. McDonald, 2018) introduced *Code Wizard*, a visualization tool embedded in Excel that leverages the certainty level of the codes assigned by all coders to highlight highly ambiguous codes. In addition, it enables the aggregation of individual coding tables, the automated sorting and comparison of coded data, and the calculation of inter-rater reliability. All the above works improve discussions between coders, allowing coders to focus on the most challenging aspects of the work. However, they caused the CQA process to diverge somewhat from the traditional coding procedure, introducing additional steps and consequently increasing the overall complexity (Chinh et al., 2019). As a consequence, these tools demand much learning and acclimatization, which could potentially pose new challenges for users.

On the other hand, current commercial CQA software like MaxQDA[3], Atlas.ti[4], nVivo[5] and Google Docs (Freitas et al., 2017; Nielsen, 2018), demonstrate a more intuitive and user-friendly approach, largely maintaining the familiarity of traditional coding procedures while offering the benefits of modern collaborative tools. MaxQDA, for instance, provides a feature for merging coding documents from multiple coders once independent coding is complete (*Secure & Seamless Cloud Collaboration for Teams* n.d.; Oswald, 2019; Marathe and Toyama, 2018b). The web version of Atlas.ti and Google Docs boasts even more advanced capabilities: they permit multiple users to code simultaneously and synchronize modifications in real time. The seamless integration of simultaneous coding and real-time synchronization significantly reduces workflow disruptions and promotes efficient and effective collaboration, addressing the issue of delayed information updates among team members, a common drawback found in other systems (Marathe and Toyama, 2018b). However, we've noticed that tools like Atlas.ti and Google Docs allow users to code and view others' codes within a shared document. This functionality could potentially introduce substantial bias among coders (Anderson, Guerreiro, and J. Smith, 2016), as it means that a coder's work is continually visible to their peers.

Meanwhile, there's a growing interest in leveraging AI to augment CQA. For example, Rietz et al. (Rietz and Maedche, 2021) recognized the potential of AI in CQA and call for efforts to examine the extent to which code rules and formulas can assist multiple coders in discussing their interpretations of codes during the coding process. The

[3]https://www.maxqda.com

[4]https://atlasti.com

[5]https://lumivero.com/products/nvivo/nvivo-product-tour/

authors also suggested investigating how coders interact with suggestions generated based on their co-coder's coding rules. This underscores the prospective advantages that AI integration could introduce to the CQA process.

In summary, although collaboration is fundamental in qualitative analysis, and AI has been extensively explored in individual qualitative analysis contexts, the realm of **AI-assisted collaborative qualitative analysis** is still relatively underexplored. This gap presents a unique opportunity to investigate the feasibility and potential of AI in enhancing collaborative qualitative analysis at that stage.

3.3 Generative AI and Qualitative Analysis

Since the end of 2022, ChatGPT[6], a chat interface powered by OpenAI's LLMs like GPT-3.5, has revolutionized human-computer interaction. This innovation has ushered in a new era of human-AI interaction and human-LLM interaction. Overnight, human-centered AI and human-AI collaboration have become the focal points of HCI research.

3.3.1 Large Language Models and Generative AI

There have been two significant shifts in the learning approaches of NLP models (P. Liu et al., 2023). The first occurred between 2017 and 2019, marking a departure from the earlier supervised learning paradigm, where task-specific models were trained exclusively on datasets of input-output examples for targeted tasks, such as text classification. Post-2017-2019, the standard practice transitioned to the 'pre-train and fine-tune' paradigm. Here, a model with a fixed architecture is initially pre-trained as a language model (LM).

Around 2021, the second sea change emerged, replacing the 'pre-train, fine-tune' approach with a 'pre-train, prompt, and predict' method. Instead of adapting pre-trained LMs to downstream tasks through objective engineering, this new approach reformulates downstream tasks to resemble those solved during the original LM training, utilizing textual prompts. This method enables the use of a single LM, trained in an entirely unsupervised manner, to solve a wide array of tasks using a suite of appropriate prompts. Early models that exemplify these shifts include GPT, GPT-2, and GPT-3.

These models are also known as Generative AI. IBM describes it as: "Generative AI refers to deep-learning models that can generate high-quality text, images, and other content based on the data they were trained on."[7]. Given this capability, Generative AI has found applications in various fields, including education, gaming, the metaverse, media, advertising, film, music, and coding, to name a few (Chaoning Zhang et al.,

[6]https://openai.com/blog/chatgpt
[7]https://research.ibm.com/blog/what-is-generative-AI

2023). In education, for instance, generative AI aids teachers in designing course materials and providing personalized tutoring. In the media sector, it enhances the efficiency of practitioners through tools like automated writing assistants, virtual news anchors, and caption generation, among others (Xiao, 2023). Each area also benefits uniquely from the technology's ability to create and innovate, thereby shaping new possibilities and experiences.

3.3.2 Generative AI and Human-LLM Collaboration

There are numerous methods to apply generative AI across different fields. For designers, one approach is to create various interfaces or tools tailored to specific tasks. Writing, specifically, has gained popularity as a task due to its alignment with the input and output capabilities of LLMs in text form. Researchers have explored using LLMs to support writing, ideation, storytelling, and more. For instance, Lee et al.(M. Lee, Liang, and Yang, 2022) introduced CoAuthor, a human-AI collaborative writing dataset that demonstrates language, ideation, and collaboration capabilities in creative and argumentative writing. Building on this, the work of Zhang et al.(Z. Zhang et al., 2023) delved into the chain-of-thought interface design (Wu, Terry, and Cai, 2022; Wu, E. Jiang, et al., 2022), which aids in the ideation of arguments. This design enables users to develop arguments from a single point to multiple points and to generate text with varying levels of abstraction. Similarly, Kim et al. (Kim et al., 2023) introduced a design framework that aids in organizing both the input and output for generative AI tasks. This framework enables interface designers to empower users in generating various iterations of textual objects. It supports the flexible creation, modification, and linking of objects, allowing users to experiment with diverse configurations.

3.3.3 Using Generative AI to Support Qualitative Analysis

Recent advancements in LLMs also have sparked interest in their application for qualitative analysis, owing to their enhanced text generation, comprehension, and summarization abilities (OpenAI, 2023). To enhance coding efficiency, Atlas.ti Web has incorporated OpenAI's GPT models to provide one-click code generation and AI-assisted code suggestions[8]. Other software predominantly depend on manual human evaluation or basic AI applications, such as word frequency counting or sentiment analysis.

On the research side, Byun et al. (Byun, Vasicek, and Seppi, 2023) posed the question: *"Can a model possess experiences and utilize them to interpret data?"* They examined various prompts to assess theme generation by models such as *text-davinci-003*, a fine-tuned variant, and ChatGPT (referred to as gpt-turbo in their experiment). Their approach

[8] https://atlasti.com/atlas-ti-ai-lab-accelerating-innovation-for-data-analysis

involved methods like one-shot prompting and question-only techniques. Their findings suggested that these models are adept at producing human-like themes and posing thought-provoking questions. Furthermore, they discovered that subtle changes in the prompt — transitioning from *"important"* to *"important HCI-related"* or *"interesting HCI-related"* — yielded more nuanced results. Additionally, Xiao et al. (Xiao et al., 2023) demonstrated the viability of employing GPT-3 in conjunction with an expert-developed codebook for deductive coding. Their findings showcased a notable alignment with expert ratings on certain dimensions. Moreover, the codebook-centered approach surpassed the performance of the example-centered design. They also mentioned that transitioning from a zero-shot to a one-shot scenario profoundly altered the performance metrics of LLMs.

Chapter 4

Examining the Effectiveness of AI-assisted Human-to-Human Collaboration in Qualitative Analysis

4.1 Goals and Context

Based on our literature review, we have identified three main phases in inductive collaborative qualitative analysis, aimed at developing a coding schema. These phases include independent open coding, team discussions for forming code groups, and the final application of the coding schema. Notably, discussion is a critical phase where users resolve disagreements and form a consensus.

At the time this work was conducted, while considerable efforts had been made in AI-assisted qualitative analysis, attempts at integrating collaboration in this context were lacking. At that time, LLMs had not evolved to their current state. We utilized a pre-trained language model, Bert, exploring whether fine-tuning this model with a group's coding history could foster a shared understanding of "group knowledge."

Motivated by these, I initiated research on AI-assisted collaborative qualitative coding. Specifically, I investigated the potential of AI in supporting CQA and explored the feasibility of AI for human-to-human collaboration that goes beyond the conventional human-AI interactions found in existing work.

4.2 Formative Interview

To understand the current CQA practices, challenges, and users' anticipations when integrating AI, we conducted a series of semi-structured interviews with HCI researchers possessing varying levels of CQA experience. Our university's Institutional Review Board (IRB) granted approval for these interviews.

4.2.1 Methodology

Our semi-structured interviews involved 8 HCI researchers (4 females, 4 males, mean age = 29.9 years), all of whom possessed experience in QA and CQA. The participants encompassed two postdoctoral researchers (P1, P2), two senior graduate researchers (P3, P4) regularly utilizing (C)QA in their work, and four junior graduate researchers (P5, P6, P7, P8) possessing a minimum of 1.5 years of experience in (C)QA. Further details, including their respective fields of study, educational background, software used for (C)QA, and the duration of QA use in their research, can be found in Table 4.1.

During the interview, we asked participants to reflect upon and share their most notable CQA experiences. In addition to narrating their experiences, we requested participants to demonstrate their data coding process via screen sharing when feasible. We also explored the challenges they encountered with their chosen tools, as well as their expectations concerning potential AI assistance in the process. All interviews, lasting between 20 and 60 minutes, were conducted virtually, audio-recorded, and transcribed through Zoom to facilitate subsequent analysis.

To derive nuanced insights from our data, two authors, including the interviewer and an experienced qualitative researcher, employed a CQA process as per Richards et al. (Richards and Hemphill, 2018). The coders began with independent open coding of two representative transcripts (P2, P3). Following this initial round, they convened to discuss similarities, reconcile code conflicts, and establish consensus on a primary codebook. The codebook was then tested and refined using two additional interview transcripts (P1, P4). Lastly, the first coder processed the remaining four transcripts (P5-P8). The findings from this process are presented in the following subsection.

4.2.2 Findings

Basic CQA Process

Our participants generally followed the main steps of the CQA process outlined by Richard et al. (Richards and Hemphill, 2018). The following summarises their practices:

TABLE 4.1: Demographics of Interview Participants. Every participant was working in tje field of HCI and was a master's student or above.

Participant ID	Fields of Study	Education	QA Software	QA Experience (Years)
P1	HCI, Mobile Computing	Postdoc	Excel	9
P2	HCI, Healthcare	Postdoc	nVivo/Atlas.ti/MaxQDA/Excel	7
P3	HCI, Ubicomp	Ph.D. student	Atlas.ti/Excel	4
P4	HCI, AI	Ph.D. student	Whiteboard/Google Sheet	4
P5	HCI, VR	Ph.D. student	Word/Excel/nVivo	2
P6	HCI, Healthcare	Master's Student	Google Docs	3.5
P7	HCI, Chatbot	Master's Student	Atlas.ti	3
P8	HCI, Security, Privacy	Master's Student	Atlas.ti/Excel	1.5

1. Each coder in the team receives a copy of the data (e.g., interview transcripts, and qualitative notes). They individually review the material and identify key points and primary codes through a process known as initial coding or open coding.

2. The coders convene to discuss their respective codes and selected key points. During these discussions, they address any discrepancies or differences in understanding and interpretation of the codes. Through this, they aim to reach a consensus and propose a primary codebook. One participant described this process as follows: *"The first level of analysis is like: I have a copy of the data, and my collaborator has another copy, and we'll assign primary codes. Then we sat down and discussed them, and see if he or she agrees. If there is disagreement, we will rethink the code and discuss." (P1, postdoc with 9 years of QA experience).*

3. The coders proceed to code additional data and expand the content within the codebook. This iterative process is often repeated for several rounds until the codes stabilize. One participant explained this process as follows: *"Based on the existing [codes in] open coding, we can roughly divide it into several pieces [or groups], and then pick out a few themes. [Sometimes] it is difficult to determine the theme, then there will be several rounds discussions [to decide the themes]." (P6, Master's student with 3.5 years of QA experience).*

4. Once the codebook reaches a stable state, one or more coders employ this finalized codebook to code the remaining data.

5. The coders proceeded to generate reports based on their findings. These reports synthesized the coded data and identified emerging themes, patterns, and examples derived from the analysis.

However, in our inquiries about the specific methods employed for CQA, there were different responses regarding their preferred approaches. Some participants indicated utilizing grounded theory (J. Corbin and Strauss, 2014; Bryant and Charmaz, 2007), while others mentioned adopting thematic analysis (Maguire and Delahunt, 2017; J. A. Smith, 2015; Braun and Clarke, 2006). These methodological choices were influenced by the specific requirements and objectives of their respective projects.

Specifically, when time and resources permit, and when participants discern the primary value of their project in the findings of qualitative research—especially in the absence of solid expectations or hypotheses for the data—they typically engage in a more strict CQA process (Lazar, Feng, and Hochheiser, 2017a) or even grounded theory, as previously described. This approach fosters a deeper, more inductive, and nuanced understanding, consequently leading to insightful revelations. One participant explained the rationale behind this approach: *"I think the reason [for following a strict CQA process in this project] maybe because first, we have more people and collaborators, and* *secondly, because this paper mostly depends on the qualities of the codes we've evolved. We wish it's a primary contribution, so it's like the difference in purpose [determines the methods and the strictness levels we use]." (P4, Ph.D. student with 4 years of QA experience).*

In the context of a study that encompasses mixed contributions, participants tend to favor thematic analysis when they have clear expectations and a more structured framework in mind. This approach, encompassing steps from "Generate initial codes" to "Define themes" (Braun and Clarke, 2006; Maguire and Delahunt, 2017), places a greater emphasis on the testing and refinement of pre-defined themes than strictly following the step-by-step collaboration described above. One potential deviation observed is the utilization of a predefined codebook instead of initiating the coding process with open coding, which is a characteristic of the traditional CQA methodology. As explained by one participant: *"For me, for example, if it's research mostly employing mixed methods, meaning you have both quantitative and qualitative data, in that case, if you have the quantitative research, then you use the qualitative to support this argument from another perspective. In that case, you can directly go for something like thematic analysis, and it's easy because you know what to expect." (P3, Ph.D. student with 4 years of QA experience).*

Difficulties in Performing Collaborative Qualitative Analysis

D1: Slow and time-consuming. The participants unanimously acknowledged that CQA is a time-intensive endeavor. The complete process of qualitative coding alone can span several weeks to several months for individual coders. When collaboration is involved, the duration is further extended due to "multiple rounds" of discussion and testing a comprehensive codebook *(P4, Ph.D. student with 4 years of QA experience)*. As expressed by one participant: *"It does take a lot of time. A 30-minute interview could be converted into seven or eight thousand words. I had to spend an hour or two to do independent coding, and another three hours to discuss." (P5, Ph.D. student with 2 years of QA experience).* Moreover, in cases where the coding process extends over a significant duration, coders often find it necessary to re-read the text during the reflection stage. *"The difficulty [of CQA] could be time-consuming. Especially if you do it twice, you have to read it again." (P1, a postdoc with 9 years of QA experience).* Additionally, if one of the coders in the coding team works at a slower pace or struggles to keep up with the progress, it can result in an extension of the overall coding time *(P3, Ph.D. student with 4 years of QA experience).*

Yet, the time-consuming nature of the CQA process presents a valuable opportunity for junior researchers to actively engage and contribute, as they often have more flexibility and availability, allowing them to dedicate ample time to the rigorous coding process involved in CQA. For example, half of the participants (P1-P4) were involved in projects that fostered collaboration with junior researchers, including those with little to no prior coding experience. Remarkably, in these collaborations, the code suggestions from both groups were treated with equal importance and consideration. One participant explained their approach: *"Actually I work with some junior students. In this case, I think the first time we would encourage them to read the book chapters about how to conduct coding...After a few runs of demonstrations, in most cases, they become, you know, better QA [coders]." (P4, Ph.D. student with 4 years of QA experience)*

D2: Coding bias and the struggle for independent coding In a strict CQA process, individual coders are expected to do coding independently from their own perspectives. However, the current QA software available, such as Atlas.ti, often poses challenges in merging codes when coders perform open coding independently in separate documents. This limitation can lead to coders relying on shared documents on Atlas.ti Web or Google Docs to label the data, benefitting from their easy and real-time data synchronization feature.

However, this shared feature has its drawbacks. The visibility of each other's codes may result in mutual influence, potentially biasing the coding process. This concern was reported by 6 out of 8 participants. As one participant described: *"So the main one [difficulty] is usually very hard to separately do the coding...if one person has coded, then the next person will see that person's coding, which means that you are influenced by other coders' coding. You cannot select codes you added or switch off all the coding coming from others' perspectives. The view gets very overlapped and then gets very confusing." (P2, postdoc with 7 years of QA experience).*

Similarly, although P6 did not explicitly mention the issue of code bias, it was observed through her shared coding practices that she utilized Google Docs to collaborate with her colleagues on a shared page. This collaborative approach inadvertently allowed all collaborators to view and access the codes being generated.

D3: Trade-offs in utilizing QA software vs. traditional text editors Most participants (6/8) favored traditional text editing tools like Google Docs and Excel over professional QA software like Atlas.ti or nVivo, finding the current features of text editors sufficient for their needs. *"We use Excel because it just has all essential functions that we need, like filtering, data validation and removing duplicates." (P1, postdoc with 9 years of QA experience).*

This can also be seen as an outcome of the deliberation of weighing the learning costs against the potential benefits: QA software such as nVivo is described as having a "steep learning curve" *(P4, Ph.D. student with 4 years of QA experience)*, yet it seemingly does not provide substantial advantages over conventional tools. Users anticipate more sophisticated features in return for their significant learning investment—like auto-grouping similar codes among coders. This deliberation often leads many to favor simpler tools that are easier to master and require a lower investment in learning. One Ph.D. student with 2 years of QA experience (P5) shared, *"I also tried to use nVivo [to do* *collaboration]. I found that when using nVivo, I can not group together codes and text from three coders [automatically]. So later we just used Excel."*

However, shifting to traditional text editors can pose challenges when handling large and diverse datasets, especially with an extensive codebook. As one participant noted, *"The final codebook is very large, with a lot of themes, it is troublesome to read." (P5, Ph.D. student with 2 years QA experience).* Moreover, users may forget their proposed codes in open coding, relying on "memory and perception" (*P6, Master's student with 3.5 years of QA experience*) to generate the initial codebook. This often complicates subsequent stages of qualitative coding, requiring additional work, such as expanding the codebook or necessitating another round of CQA.

D4: Trade-offs of using a predefined codebook Some participants (3/8) indicated that the practice of lead coders proposing a primary codebook for the team to follow could streamline the coding process. As one participant described, *"I developed an initial codebook. I sent a copy to the two coders to let them do some coding on their own. Then after 1 week, they sent it back to me with their codes. Then we do a comparison to solve the disagreements." (P8, Master's student with 1.5 years of QA experience).*

Yet, this could potentially restrict the benefits of CQA. P1, a postdoc with 9 years of QA experience, shared her opinion, *"It sorts of restricts the categories. They only had to code the data in a certain way."*

However, some participants acknowledged situations where, despite recognizing it as less than "strict", they would adopt this strategy due to time constraints: *"For my project, we have very limited time to do coding, then my collaborator and I do coding for the whole transcripts with the codebook I proposed. If I have enough time, I would definitely do a stricter qualitative analysis."(P4, Ph.D. student with 4 years of QA experience).*

D5: Interpretational variations and granularity challenges Interpretational divergence is a common occurrence among coders working on the same data, leading to distinct coding outcomes. P1, a postdoc with 9 years of QA experience, stated, *"The difficulty comes after [coding] the first round codes. We always have different ways of interpreting things."*. Furthermore, coders could assign codes with differing levels of specificity, causing extra work and necessitating further discussion to reconcile the codes: *"My codes are broader, my partner is more specific. My codes usually cover the whole category. But hers give a bit more subcategories...the thing is I don't want the subcategories. I just want all the codes to be the same categories."(P7, Master's student with 3 years of QA experience).*

Suggestions for AI-assisted CQA tools

S1: AI generates code suggestions based on code history The majority of our participants (6/8) express a desire for AI to provide code suggestions, with the most preferred source being their personal coding history. *"So if this content and that content suddenly matches up, it would be nice for AI to just assist me and say, 'hey, you've done this before! Why don't you assign this code to this code that you have already assigned?' And I may not have remembered, because there were a lot of things I processed along the way." (P1, a postdoc with 9 years of QA experience).*

P3, a Ph.D. student with 3 years of QA experience, expressed skepticism towards an AI system in which codes are derived from external sources, such as other projects or AI-generated suggestions: *"Initially, the coding is a thinking process. I don't want to interrupt the [open coding] thing. I don't want to be biased by somebody in the thinking process...Because any suggestion is biased. Once I finish initial coding, AI can suggest like, for example, then AI just shows like these are similar. These are different. Then I can use that feature similar to the initial coding. Then you refer my codes to feed the AI suggestions. I think that can improve the coding. If AI suggests [by its own], I don't trust anyway."*

S2: AI facilitates coding conflict detection In our discussions with participants, we confirmed Zade et al.'s (Zade et al., 2018) previous analysis that a text selection analyzed by multiple coders could lead to 1) divergence: entirely distinct or even opposing interpretations, or 2) diversity: identical interpretations conveyed through different expressions, such as "not bad" and "well". Both scenarios necessitate coders to scrutinize the text, highlight the disparities and engage in a dialogue to reach a consensus on the final code. This process represents a significant time commitment.

To handle the "diversity" conflict, our participants (4/8) anticipate that AI can play a crucial role in automatically detecting variations in codes assigned by different coders to the same text. This would prompt them to continuously reassess their coding choices in real-time throughout the coding process. As described by P7, *"I think the collaboration is a matter of recommending what other collaborators have already done...for instance, one sentence, 'oh the chatbot did not understand me.' Then my code is 'chatbot is stupid', and the code of another coder is 'chatbot has insufficient data'. AI may detect this difference [in real time]." (P7, Master's student with 3 years of QA experience).*

Moreover, P7 further explained that this conflict detection can take place during the coding process, fostering timely discussion among coders and eliminating the necessity to recall the meaning of their own codes or engage in manual one-by-one comparisons after coding. As a result, they would only need to allocate a lower amount of time to resolve discrepancies. This is particularly true when confronting with code "divergence": *"Then the difference may be detected. AI says, 'So what's the difference? Why did you choose a* *different code? If you choose this code, can you give a reason why you choose this code?' Then I tell the other coder to come back, 'guys. Let us discuss' (P7, Master's student with 3 years of QA experience).*

4.2.3 Study Limitation

It is important to acknowledge that our findings may have limitations. One main limitation is the background of our participants, as all but one (P2) are coming from a technology-oriented background, as indicated in Table 4.1. This could potentially impact our findings, as their projects often involve mixed methods and practices that may be influenced by their technological perspectives rather than perspectives and methods rooted in social science or psychology. Our study also includes participants with varying experiences with CQA (experts and non-experts). Our goal here was to encompass a range of perspectives and experiences, allowing us to address both the learning issues associated with applying CQA and the inherent issues that arise in its implementation.

4.2.4 Discussion

In practical applications, similar CQA steps are employed by our participants with slightly different forms, but the majority of practitioners primarily adhere to the six CQA steps proposed by Richards et al. (Richards and Hemphill, 2018). Regarding the anticipated stages of AI integration, our findings support the conclusions drawn in the previous study conducted by Feuston et al. (Feuston and Brubaker, 2021), that AI can be effectively incorporated into the inductive coding process. However, unlike previous research that primarily focuses on utilizing AI for pattern identification and data interpretation at this stage, our findings demonstrate an alternative use of AI: detecting coding conflicts between coders and facilitating collaborative interpretation and evolution of data among them. Building upon the established CQA stages and the anticipated integration of AI, we have designed a study task for our evaluation of CoAIcoder consisting of three primary phases: 1) open coding, 2) codebook formation through discussion, and 3) coding using the codebook.

Our findings also confirmed several noteworthy challenges and limitations associated with CQA. These include the time-consuming nature of the process (Marathe and Toyama, 2018b; Rietz and Maedche, 2021), difficulties in achieving consensus among coders (Cornish, Gillespie, and Zittoun, 2013; Zade et al., 2018; Drouhard et al., 2017; J. A. Jiang et al., 2021; Feuston and Brubaker, 2021), and the presence of software-related issues (Hopper et al., 2021; J. A. Jiang et al., 2021; Feuston and Brubaker, 2021). In particular, we have identified a new limitation related to coding bias arising from the challenges of independent coding. This limitation is particularly observed in the current CQA software that facilitates real-time collaboration and coding on a shared document among coders. For instance, in platforms like Google Docs and Atlas.ti Web, users have visibility into each other's codes while co-coding. This visibility negatively impacts their willingness to use CQA software as it introduces a potential bias in the coding process.

In an effort to mitigate this limitation, we propose a solution that involves leveraging AI as an intermediary between the two coders. Instead of directly revealing each other's codes, our system retrieves and analyzes the coded data from both coders in real time. Incorporating the distinct perspectives of each coder, our AI system generates code suggestions that serve as indirect indicators of coding differences, which facilitate awareness of differences among the coders. This awareness encourages them to reconsider and reflect upon their own codes during the coding process, rather than solely during post-coding discussions. By doing this, we aim to minimize bias in the existing coding process when using traditional real-time collaboration software such as Google Docs, Atlas.ti Web, and others.

4.3 CoAIcoder: System Design

Based on the insights from the interviews, we have proceeded to design and implement a platform that harnesses the power of AI to facilitate the CQA process.

4.3.1 Design Considerations

1. Independence and convenience: To keep both independence and convenience (D2), each coder is assigned a separate web page with the same data, but allows access to others' coding results by simply clicking a link rather than navigating through the cumbersome process of exporting and importing that current non-web software necessitates (Marathe and Toyama, 2018b).

2. Fast learning curve: To mask the complexity of AI and flatten the learning curve (D3), our system is designed to emulate familiar platforms like Google Docs, specifically its "comment" function. This design enables the addition and removal of codes in a way that users can readily comprehend and navigate. Moreover, the web page or coding interface allows data selection at various granularity levels for code addition, similar to existing commercial QA software.

3. List of code suggestions: Our approach to collecting coding history and generating code suggestions for coders has the potential to yield significant time savings and reduce manual effort (D1). In accordance with Rietz et al. (Rietz and Maedche, 2021), our approach includes providing multiple AI suggestions, each accompanied by a confidence level. This aims to bring attention to the inherent uncertainty of codes

and curtails the potential risk of thoughtlessly adopting these suggestions (N.-c. Chen et al., 2016; J. A. Jiang et al., 2021; N.-C. Chen et al., 2018).

4. User autonomy: Code suggestions are revealed only upon user request, emulating the natural coding process. This fosters active thinking before viewing suggestions, minimizing the risk of being unintentionally guided in unwanted directions (J. A. Jiang et al., 2021) by direct exposure to others' codes (D2). This is beneficial as it encourages independent thinking, a tool to combat groupthink (Janis, 2008).

5. Collaboration: The coding histories of both coders are comparatively analyzed and subsequently input into an AI classification model. This model then generates a range of AI suggestions for coders upon request (S1). It not only provides a reference point to users based on their own coding history but also integrates their partner's history. Through this design, our goal is to foster awareness of disparities among coders, thus enhancing their understanding and reflection of the data. This becomes particularly beneficial when addressing coding styles or logic conflicts (Akpınar, Erol, and Aydoğdu, 2009) within a group.

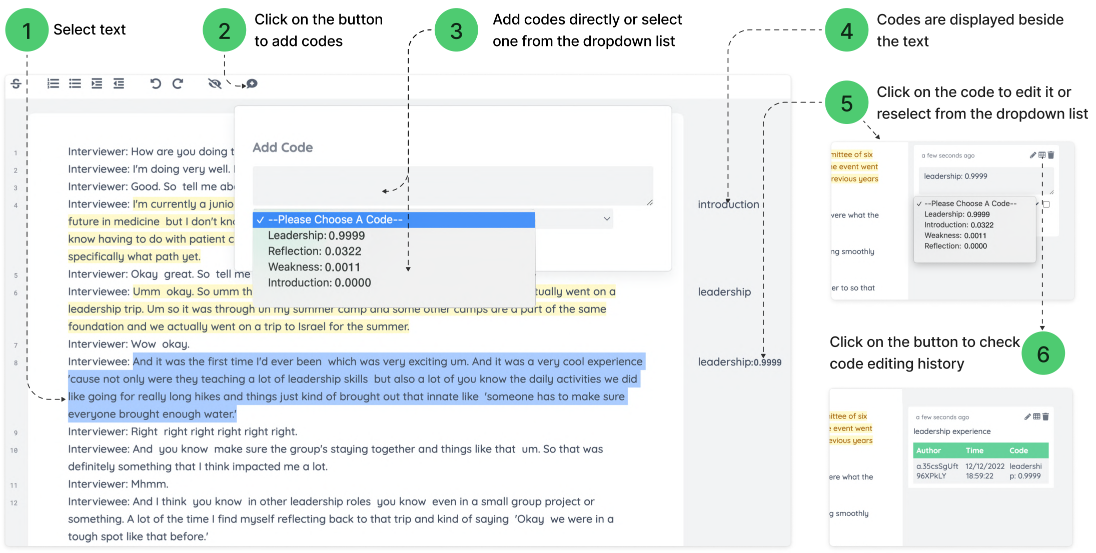
6. Usage in inductive coding: Beyond the system, instead of enforcing a rigid, predefined codebook that could potentially constrain coders (D4), we place importance on the dynamic nature of the inductive coding process in the evaluation.

The specific design components of our prototype, CoAIcoder, are elaborated in the following subsections.

4.3.2 Interface

The interface (see Figure 5.1) is built on two components: 1) *Etherpad*[1], an open-source collaborative text editor that supports multiple users editing text in real-time (Amiryousefi et al., 2021; Bebermeier and Kerkhoff, 2019; Goldman, Little, and R. C. Miller, 2011), and 2) its plugin, *ep_comment_pages*[2], which allows for adding comments beside the text. To create a code, users select the text of significance → click on the "comment" button → type in the code OR select from a list of AI-generated code suggestions → press "save". The interface also provides features for coders to review the code history, re-edit, and delete the code in case they change their minds. Additionally, the customized version of *ep_comment_pages* offers code suggestion lists containing a maximum of five codes when requested by the user. These codes are ranked by their confidence level, which ranges from 0 to 1.

[1] https://etherpad.org/
[2] https://github.com/ether/ep_comments_page


1 Select text
2 Click on the button to add codes
3 Add codes directly or select one from the dropdown list
4 Codes are displayed beside the text
5 Click on the code to edit it or reselect from the dropdown list
6 Click on the button to check code editing history
Add Code
✓ --Please Choose A Code--
Leadership: 0.9999
Reflection: 0.0322
Weakness: 0.0011
Introduction: 0.0000
introduction
leadership
leadership:0.9999
a few seconds ago
leadership: 0.9999
✓ --Please Choose A Code--
Leadership: 0.9999
Introduction: 0.0322
Weakness: 0.0011
Reflection: 0.0000
a few seconds ago
leadership experience

Interviewer: How are you doing t
Interviewee: I'm doing very well. I
Interviewer: Good. So tell me ab
Interviewee: I'm currently a junio
future in medicine but I don't kn
know having to do with patient c
specifically what path yet.
Interviewer: Okay great. So tell me
Interviewee: Umm okay. So umm th … tually went on a
leadership trip. Um so it was through … part of the same
foundation and we actually went on a trip to Israel for the summer.
Interviewer: Wow okay.
Interviewee: And it was the first time I'd ever been which was very exciting um. And it was a very cool experience 'cause not only were they teaching a lot of leadership skills but also a lot of you know the daily activities we did like going for really long hikes and things just kind of brought out that innate like 'someone has to make sure everyone brought enough water.'
Interviewer: Right right right right right right.
Interviewee: And you know make sure the group's staying together and things like that um. So that was definitely something that I think impacted me a lot.
Interviewer: Mhmm.
Interviewee: And I think you know in other leadership roles you know even in a small group project or something. A lot of the time I find myself reflecting back to that trip and kind of saying 'Okay we were in a tough spot like that before.'

Author	Time	Code
a.35csSgUft 96XPkLY	12/12/2022 18:59:22	leadership: 0.9999

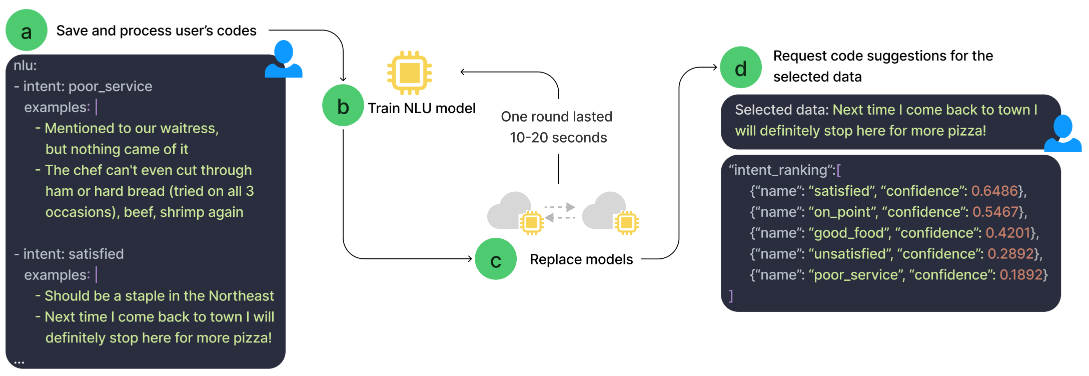
FIGURE 4.1: The CoAIcoder interface. Creating a code: (1) Users select the text of significance and (2) click the comment button to add codes. (3) Users can add codes directly or select one from the dropdown list. (4) The created code is shown beside the selected text. (5) Click on the code to edit it directly or reselect codes. (6) Click on the button to check the code editing history.

4.3.3 AI Model

The AI model harnesses the Rasa NLU framework[3], to generate code suggestions upon a user request. Previously, Rasa (Bocklisch et al., 2017; J. Cao et al., 2023) has been deployed in the domain of HCI for handling conversations in several prototype models (Porfirio et al., 2019). Specifically, we employ a recommended NLU pipeline from Rasa to train an NLU classification model[4]. This pipeline incorporates multiple components: `SpacyNLP`, `SpacyTokenizer`, `SpacyFeaturizer`, `RegexFeaturizer`, `LexicalSyntacticFeaturizer`, two instances of `CountVectorsFeaturizer`, and `DIETClassifier`.

Within this pipeline, the SpacyNLP language model 'en_core_web_sm'[5] is selected for training efficiency consideration, comparing to larger pre-trained language models utilized in the SpacyFeaturizer[6]. Moreover, the DIET (Dual Intent and Entity Transformer) Classifier (Bunk et al., 2020) is selected for its capability to perform multi-class classification. The NLU pipeline operates on a computer equipped with Ubuntu 20.04, Tensorflow

[3]https://rasa.com/docs/rasa/
[4]https://rasa.com/docs/rasa/tuning-your-model/#configuring-tensorflow
[5]https://spacy.io/usage/models
[6]It should be noted that our coding materials in the evaluation consist solely of simulated job interview transcripts and do not encompass any specialized domain knowledge.


a
Save and process user's codes
b
Train NLU model
One round lasted
10-20 seconds
c
Replace models
d
Request code suggestions for the selected data

```
nlu:
- intent: poor_service
  examples: |
    - Mentioned to our waitress,
      but nothing came of it
    - The chef can't even cut through
      ham or hard bread (tried on all 3
      occasions), beef, shrimp again

- intent: satisfied
  examples: |
    - Should be a staple in the Northeast
    - Next time I come back to town I will
      definitely stop here for more pizza!
...
```

Selected data: Next time I come back to town I will definitely stop here for more pizza!

```
"intent_ranking":[
    {"name": "satisfied", "confidence": 0.6486},
    {"name": "on_point", "confidence": 0.5467},
    {"name": "good_food", "confidence": 0.4201},
    {"name": "unsatisfied", "confidence": 0.2892},
    {"name": "poor_service", "confidence": 0.1892}
]
```

FIGURE 4.2: The pipeline for training and updating the CoAIcoder model, designed to facilitate code suggestion requests. (a) Save and process user's coding data: This step involves saving, retrieving, and processing each coder's coding data. The retrieved data is then used to generate an *nlu.yml* file, which contains coded text (i.e., examples) and codes (i.e., intent). (b) Train NLU Model: This process trains a new model using the updated *nlu.yml*, which takes about six seconds or longer, depending on the coding data size. (c) Replace models: This process substitutes the old model with the newly trained one, approximately requiring four seconds per model. (d) Request code suggestions: The user requests code suggestions from the server. Initially, CoAIcoder requests code suggestions from server1. If this fails, the request is then rerouted to server2, thereby sustaining the impression of continuously updated code suggestions for the user.

(2.6.1), CUDA (11.2), and an Nvidia GPU Quadro K2200 graphics card, in conjunction with installed software like Rasa (3.0), Node.js (17.2.0), and MongoDB (5.0.4).

The AI model, trained on users' coding histories, may not provide suggestions for the initial few requests due to a lack of historical data. However, as the data pool expands, it gains the capability to generate up to five code recommendations for each request, sorted according to their respective confidence levels. The DIET Classifier computes these confidence levels, indicating the cosine similarity score between predicted labels and text[7].

4.3.4 Training and Updating Pipeline

The training and updating pipeline is shown in Figure 4.2.

Saving and Retrieving Data

Each user's coding data is individually stored in the database. To identify conflicts, the codes of each coder are compared with both their own and their peers' coding histories. The codes are subsequently grouped and deduplicated, readying them as inputs for

[7] https://rasa.com/docs/rasa/components/#dietclassifier

the NLU pipeline. For example, if two sentences are labeled with the same code, they are grouped into one "intent" (akin to the "class" concept in Machine Learning, and equivalent to "code" in this work) in the Rasa NLU data file, *nlu.yml*. If a single sentence is coded with two distinct codes by two coders, it serves as an "example" for both "intent" in the Rasa NLU file. If two codes convey similar meanings but have different expressions, they are interpreted as two separate "intent" (in the current version of CoAIcoder). It should be noted that this process may slightly vary under the four conditions outlined in Section 4.4.

Training and Reloading NLU Models in (Near) Real Time

Firstly, the *nlu.yml* file is fed into the NLU pipeline for automatic training of the new NLU model. Secondly, the trained model is promptly uploaded to the Rasa server via HTTP API, replacing the previous model. Lastly, we configure two Rasa Open Source servers to run on ports 5005 and 5000, respectively. These servers act as buffers for user requests, utilizing server swapping. Users have the capability to request code suggestions through HTTP from either of the two servers. In case CoAIcoder fails to receive a response from one server due to the server's ongoing model update process, it swiftly switches to the other server as an alternative. The complete pipeline typically requires approximately 10 to 20 seconds or longer, depending on the size of the coding data. Users experience a seamless process without any interruptions caused by model updates.

4.4 User Evaluation Design

We proposed four collaboration methods for using CoAIcoder (see Figure 4.4). To ensure a fair evaluation, we focus on novice users who primarily participate in the CQA process (see section 4.2.2) and undergo requisite coding training. Moreover, the similar starting point of the participants facilitated a balanced evaluation of CoAIcoder using a between-subject design. This user study has been approved by our university's IRB.

4.4.1 Task

We simulated a CQA task encompassing three main steps: independent and open coding, codebook formation through discussion, and codebook application. Due to the multiple factors involved in our study, we simplified the study by focusing on the CQA process involving two users. To determine the optimal duration for each phase and the overall study, we conducted a pilot test with eight participants (5 females, 3 males, mean age = 24.7).

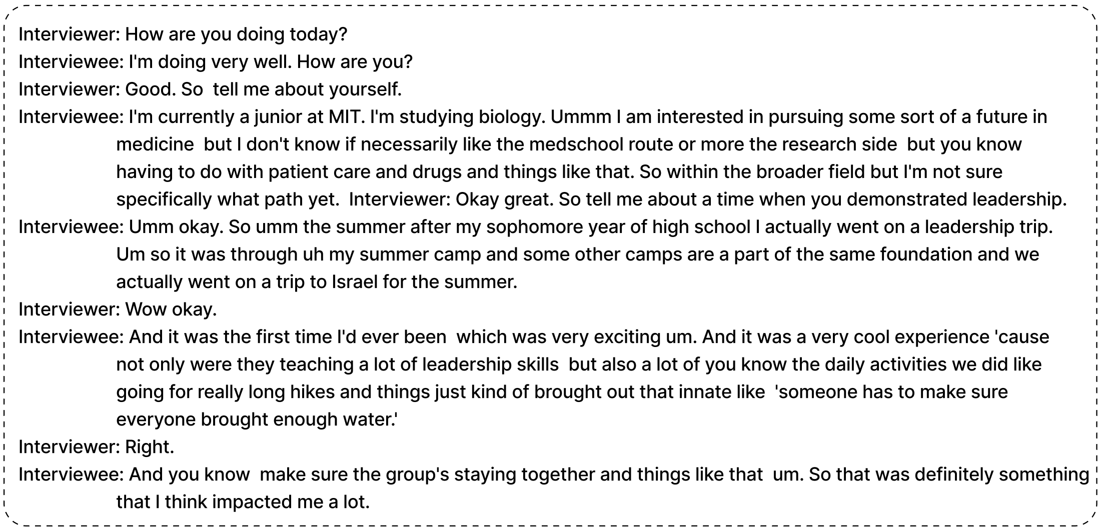

Interviewer: How are you doing today?
Interviewee: I'm doing very well. How are you?
Interviewer: Good. So tell me about yourself.
Interviewee: I'm currently a junior at MIT. I'm studying biology. Ummm I am interested in pursuing some sort of a future in medicine but I don't know if necessarily like the medschool route or more the research side but you know having to do with patient care and drugs and things like that. So within the broader field but I'm not sure specifically what path yet. Interviewer: Okay great. So tell me about a time when you demonstrated leadership.
Interviewee: Umm okay. So umm the summer after my sophomore year of high school I actually went on a leadership trip. Um so it was through uh my summer camp and some other camps are a part of the same foundation and we actually went on a trip to Israel for the summer.
Interviewer: Wow okay.
Interviewee: And it was the first time I'd ever been which was very exciting um. And it was a very cool experience 'cause not only were they teaching a lot of leadership skills but also a lot of you know the daily activities we did like going for really long hikes and things just kind of brought out that innate like 'someone has to make sure everyone brought enough water.'
Interviewer: Right.
Interviewee: And you know make sure the group's staying together and things like that um. So that was definitely something that I think impacted me a lot.

FIGURE 4.3: One sample interview transcript in coding task.

We selected mock interview transcripts from an open-source dataset (Naim et al., 2015) as the text materials to be coded in our study. This dataset covers various general topics, including leadership, personal weaknesses, and challenging experiences, which are familiar and accessible to most users. To ensure control over potential effects on both the AI model and user understanding, we specifically chose three transcripts that exhibit better clarity and coherence, each consisting of approximately 1000 words. Part of a sample interview transcript is provided in Figure 4.3.

4.4.2 Independent Variables (IVs) and Conditions

To understand the impact of AI on human-to-human collaboration in CQA, we identified three factors for our study:

1. AI {With, Without}: Whether or not AI (the NLU model) is applied to provide code suggestions. This IV aims to understand whether the use of AI affects CQA performance.

2. Synchrony {Synchronous, Asynchronous}: Whether or not two coders do coding in real-time and simultaneously. This IV aims to understand whether CQA performance is achieved similarly in synchronous and asynchronous modes.

3. Shared Model {With, Without}: Whether or not two coders use a shared NLU model to request AI suggestions. The shared model can collect coding history from both coders and be trained on it to provide code suggestions. Without a Shared model, each coder can only get AI suggestions based on his/her own coding history. This IV aims to understand if a shared model affects the CQA performance.

We combined these factors and removed meaningless and duplicate combinations, resulting in four final conditions (A-D) for the collaboration (see Table 4.2). The final conditions are shown in Figure 4.4.

TABLE 4.2: The combinations of the three factors, which are not entirely independent of each other. For instance, the absence of AI renders the Shared Model factor irrelevant, making conditions C7 and C8 nonsensical. Synchronous coding is only applicable in the presence of a Shared Model, as the order in which coding occurs is inconsequential without a shared model. Thus, certain conditions become identical due to the absence of a shared model, namely C1 and C2, as well as C3 and C4.

Combination	C1	C2	C3	C4	C5	C6	C7	C8
AI	×	×	✓	✓	✓	✓	×	×
Synchrony	×	✓	✓	×	×	✓	×	✓
Shared Model	×	×	×	×	✓	✓	✓	✓
Condition	A	A	B	B	C	D	×	×

Condition A: Without AI, Asynchronous, not Shared Model (Traditional Coding): In Phase 1, each coder independently applies codes to the first two interview transcripts using distinct web pages. In Phase 2, the coders convene to establish a shared codebook. During Phase 3, they independently apply codes to the third transcript based on the codebook. As new codes are entered, the model undergoes automatic updates.

Condition B: With AI, Asynchronous, not Shared Model: In Phase 1, each coder independently applies codes to the first two interview transcripts, each utilizing an individual model on separate web pages. These two independent models are automatically trained during the coding process, with the training data sourced from their individual code histories. In Phase 2, the coders convene to formulate a shared codebook. During Phase 3, they independently apply codes to the third transcript based on the codebook, with the model undergoing automatic updates as new codes are entered.

Condition C: With AI, Asynchronous, Shared Model: In Phase 1, coder1 independently assigns codes to the first two interview transcripts. The model is concurrently trained and offers real-time code suggestions as the coding process unfolds. Once coder1 completes the task, coder2 commences coding on a separate web page. Initially, the code suggestions are entered by coder1's coding history. Over time, as coder2's codes are introduced, they are incorporated into the suggested codes. In Phase 2, the coders collaborate to create a shared codebook. During Phase 3, they independently assign codes to the third transcript using the codebook, triggering automatic updates of the model as new codes are inputted.

Condition D: With AI, Synchronous, Shared Model: In Phase 1, both coders independently assign codes to the first two interview transcripts, each working on a separate web page. As the coding process progresses, CoAIcoder accesses their individual code

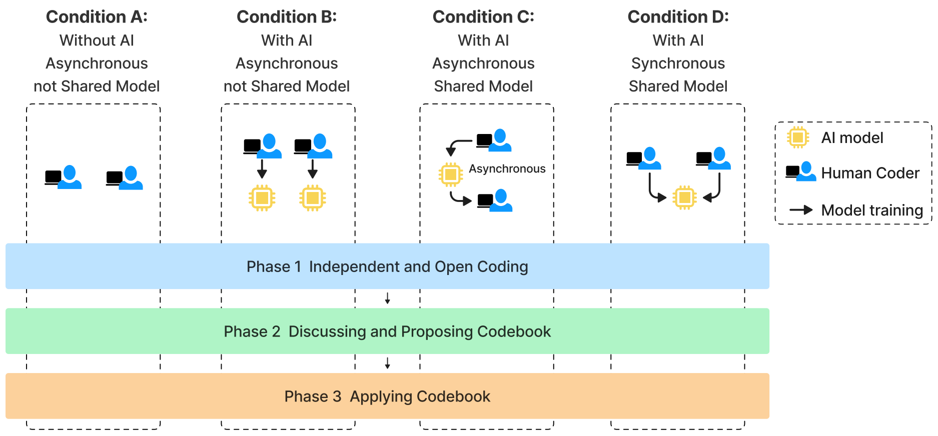


FIGURE 4.4: Four Approaches to Collaboration in Qualitative Analysis. *Condition A: Without AI, Asynchronous, not Shared Model* (Traditional Coding): Both coders independently apply codes. *Condition B: With AI, Asynchronous, not Shared Model*: Each coder applies codes using their respective NLU models. *Condition C: With AI, Asynchronous, Shared Model*: The coders apply codes asynchronously with a shared NLP model. Coder1 begins the process, during which the NLP model trains and offers real-time AI suggestions. Once Coder1 completes the task, Coder2 commences with coding. *Condition D: With AI, Synchronous, Shared Model*: The coders apply codes synchronously with a shared NLP model.

histories and automatically trains the model 3-6 times per minute. Upon request, it then provides code suggestions for both coders. In Phase 2, the coders collaborate to develop a shared codebook. During Phase 3, they independently apply codes to the third transcript using the codebook. The model is automatically updated when new codes are inputted.

4.4.3 Participants

A total of 64 participants (41 females, 23 males), ranging from 18 to 57 years old (mean=25.3, median=23), were recruited through our university's email system and public channels on Telegram targeting other universities within Singapore. All participants were native English speakers and reported no prior experience in qualitative analysis. In accordance with our university and national guidelines for participant reimbursement, each participant received an hourly compensation of 10 Singapore Dollars.

4.4.4 Procedure

The final study process was structured based on the setup depicted in Figure 4.5. A total of 64 participants were divided into 32 pairs, with each study group comprising 8

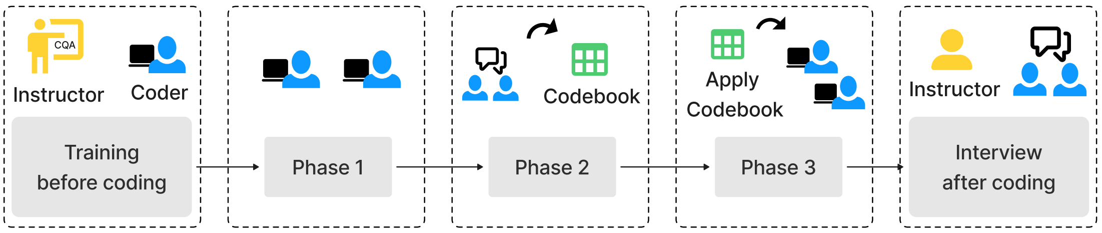


FIGURE 4.5: Study procedure. Both coders underwent training in CQA, prior to the formal coding process. **Phase 1 (Independent and Open Coding)**: In this phase, both coders individually performed coding for two interview materials, following the assigned study setup ($\leq$20 minutes). **Phase 2 (Discussion and Codebook Formation)**: During this phase, the two coders engaged in discussions to collaboratively create a structured codebook using Google Sheets ($\leq$40 minutes). **Phase 3 (Application of the Codebook)**: In this phase, the coders independently applied the codes from the agreed codebook during their individual coding sessions ($\leq$10 minutes). At the end of each phase, participants were required to complete a survey and interview ($\approx$5 minutes).

pairs. The allocation of participants to the four study groups was random, corresponding to the four conditions specified in Figure 4.4.

To ensure participants had a thorough understanding of the task, we conducted an approximately 15-minute training session prior to the study. The training session consisted of two key components:

1. Instruction: Participants received instruction from an instructor's explanation that demonstrated how to use CoAIcoder for selecting the text, adding codes in Phase 1, creating a codebook in Phase 2, and applying it in Phase 3.

2. Training Tutorial and Q&A: Participants were shown a video explaining the principles of qualitative coding, and they also received a written tutorial explaining the concepts of qualitative coding, codes, subcodes, and accompanying examples. A question and answer session was held to address any queries and ensure clarity.

Following the training session, participants engaged in the three phases as illustrated in Figure 4.5. After the study, we also gathered demographics information and invited participants to provide feedback through an interview regarding their experience with the system. The interview encompassed questions about their experience using the software, focusing on individual and collaborative coding challenges, as well as their attitude towards CoAIcoder features. With the participants' consent, we made audio recordings of the post-interviews to facilitate subsequent analysis. This analysis allowed us to gather valuable insights and perspectives on the user experience of the tasks.

In addition, we implemented the following measures to ensure the tasks were performed effectively:

1. Time reminders: Regular reminders were provided to participants to keep them aware of the remaining time. For example, notifications were given when there were 15 minutes left and 5 minutes left in the allotted time frame.
2. Coding quality check: We monitored their codes and selected text to address any questions, issues, or potential misunderstandings that participants may have encountered during the task. However, we made a conscious effort to minimally interfere with their coding process to maintain the integrity of their work.
3. Encouragement for diverse codes: Actively encouraged participants to generate a wide range of diverse codes during the coding process.

4.4.5 Dependent Variables (DVs)

Coding Time

This DV quantifies the time taken for each of the three phases of the study (see Figure 4.5). An approximate 90-minute time limit was enforced to regulate the study duration for each pair. Specifically, the time allotment for each phase is as follows: Training: $\approx$ 15 minutes; Phase 1: $\leq$ 20 minutes; Phase 2: $\leq$ 40 minutes; Phase 3: $\leq$ 10 minutes; Post survey and interview: $\approx$ 5 minutes. This structure was established in light of our pilot studies, during which we observed that most participants were able to satisfactorily complete the coding task. However, the actual coding time usage may deviate from the estimate. We assessed the actual time used in each phase for further analysis.

Inter-rater reliability (IRR)

IRR is a metric that gauges the level of consensus among multiple coders (Kurasaki, 2000). In our study, we specifically employ Cohen's kappa (κ), a measure designed to compute agreement between two coders (McHugh, 2012). The computation of κ encompasses both Phase 1 and Phase 3.

Code Diversity

This DV quantifies the diversity of proposed codes, a factor that is intimately linked to coding quality. To measure this, we count the number of *unique* codes—also referred to as first-level codes—and subcodes, or second-level codes. Here, "unique" signifies that variations of a similar meaning are counted as a single code.

Code Coverage

This DV quantifies the degree of overlap between the individual coders' codebooks and the merged codebook, at both the first and second coding levels. This applies to the initial codebook from Phase 1 and the users' proposed codebook in Phase 2. The merged codebook is iteratively developed by consolidating common codes from the 32 codebooks created during Phase 2.

The formula used for the calculation is:
$Code\ Coverage = \frac{|\,Coders'\ Codebook \cap Merged\ Codebook\,|}{|\,Merged\ Codebook\,|}$,
where $\cap$ represents the intersection of the two codebooks, and "| |" signifies the number of codes.

4.4.6 Data Analysis

Step 1: Data Integrity and Quality Checking

Following data collection, our next step was to verify the integrity and quality of the collected data:

1. 85% Completion Rate: Participants should complete coding for more than 85% of the provided data within the given time[8];

2. Task Correctness and Active Collaboration: We examined whether participants performed tasks correctly and collaborated actively. Instances of extremely low code diversity, such as using a single broad code like "Experience" to describe all examples in the interview transcripts, were a cause for concern. This lack of diversity suggested an inability to form a quality codebook. Additionally, pairs that were not willing or able to engage in productive discussion, choosing instead to develop their own individual codebooks, were noted.

Four pairs that did not meet these two criteria were omitted and replaced with new participants to maintain the data integrity and quality.

Step 2: Generating Initial Codebooks

Phase 1 and Phase 2 represent the "pre-discussion" and "post-discussion" stages, respectively. In the pre-discussion stage, the codes present a higher degree of variation. Following the discussion stage, these varied codes have been deliberated and consolidated, with differing variants merged.

In order for us to assess the IRR, code diversity and code coverage, two authors then manually formed initial codebooks for each pair. These codebooks were specifically designated for assessing the aforementioned DVs, and were neither shared with nor used by the participants at any stage during the experiment. They first coordinated to

[8]Our time regulations were established based on pilot studies, during which most native speaker participants were able to complete the coding tasks. This is designed to prevent the study from becoming overly lengthy, which could lead to fatigue and a subsequent loss of focus and motivation.

merge codes of similar meaning, adhering to the following criteria and steps: codes conveying similar meanings but expressed differently were treated as second-level codes; subsequently, the authors collaborated to propose a corresponding first-level code for each central meaning in the initial codebook. For example, "`Introduction of Leadership Experience`", "`Description of Leadership Experience`", and "`Application of Leadership`" serve as three second-level codes that fall under one first-level code: `Leadership`. For more details, please refer to Table 4.3.

TABLE 4.3: Part of a sample of the initial codebook (Phase 1). Each row containing second-level codes is counted as a single first-level code. This codebook demonstrates a "Code Diversity" of 5 first-level codes alongside 10 second-level codes.

First-level Code	**Second-level Code**		
Career Goal	Personal introduction and future (career) goals	Choosing of (academic and career) route	Not very sure about future (career) goal
Personal Interest	Personal introduction and interest area.	Interest in oil fossil fuels	
Leadership	Introduction of leadership experience	Description of leadership experience.	Application of leadership
Teamwork	Intro of working with team on big project		
Initiative	Shows initiative		

TABLE 4.4: Part of a sample codebook, which is formulated from the 27-minute discussion in Phase 2 between participants P27 and P28 under *Condition D: With AI, Synchronous, Shared Model*. The "First-level Code" column represents the first-level codes generated during this discussion. The "Second-level Code" column, on the other hand, contains codes proposed by them in Phase 1. The "Example" column showcases selected segments of the original text.

First-level Code	**Second-level Code**	**Example**
Interest and goals	Uncertain about the future	Ummm I am interested in pursuing some sort of a future in medicine but I don't know if necessarily like the med school route or more the research side but you know having to do with patient care and drugs and things like that. So within the broader field but I'm not sure specifically what path yet
		I think for me where I'm where I am at this point where I'm deciding between sort of going the medical school route or the research route;
	Show interest in alternative energy	I'm very interested in energy applications so um from alternative energy to more traditional sources so basically oil and fossil fuels. Um and kind of optimizing that industry I think there's a lot of potentials there so that's where my main interest is.

Step 3: Measuring DVs

We evaluated various DVs, including *Coding Time*, *IRR*, *Code Diversity*, and *Code Coverage*, throughout the three phases.

In terms of *Coding Time*, we concluded each phase as soon as coders exceeded the time limit. All but four pairs completed the coding task within this limit – two in *Condition B: With AI, Asynchronous, not Shared Model*, one in *Condition C: With AI, Asynchronous, Shared Model*, and one in *Condition D: With AI, Synchronous, Shared Model*.

However, we still consider these four groups as "completed" due to their achievement of a minimum 85% completion rate.

For *IRR*, we segmented the complete interview data into sentences $\rightarrow$ represented codes numerically as "0" (for sentences without codes), "1", "2", "3", etc. $\rightarrow$ Cohen's Kappa (κ) was calculated for first-level codes in Phases 1 and 3, as Phase 2 was a discussion session without new coding data. Due to the considerable variability, second-level codes were excluded from IRR computation as it was infeasible.

For *Code Diversity* and *Code Coverage*, both first-level and second-level codes from the initial codebook and the proposed codebook were incorporated into the computing process.

Moreover, we conducted a thematic analysis (Braun and Clarke, 2006; Maguire and Delahunt, 2017) for the interview audio transcripts, given that most data align with the structure provided by the interview questions.

Step 4: Statistical Analysis

We conducted a non-parametric analysis on the quantitative data (*Coding Time*, *IRR*), given concerns about the normality of the distribution of the data collected. Consequently, Kruskal-Wallis test was employed to identify the main effect, and Mann-Whitney U-test was used for pairwise comparisons.

4.5 Quantitative Results

4.5.1 Coding Time

Total Time

The average total time for each study group and the average used time for three phases is shown in Figure 4.6 and 4.7. A Kruskal-Wallis test **did not reveal any main effect** of the conditions on the total time to complete the study ($\chi^2_{(3)} = 6.71, p = .082$). In general, *Condition D: With AI, Synchronous, Shared Model* was the fastest ($M = 46.56\ mins$), followed by *Condition C: With AI, Asynchronous, Shared Model* ($M = 49.38\ mins$), *Condition B: With AI, Asynchronous, not Shared Model* ($M = 54.69\ mins$) and *Condition A: Without AI, Asynchronous, not Shared Model* ($M = 57.31\ mins$). The time for individual phases is however more informative to understand the potential effect of AI on CQA performance, therefore, we also analyzed the time for each phase.

Phase 1

We found **a significant main effect** of the conditions in Phase 1 ($\chi^2_{(3)} = 9.03$, $p = .028$). A post-hoc pairwise comparison with a Mann–Whitney U-Test shows that the

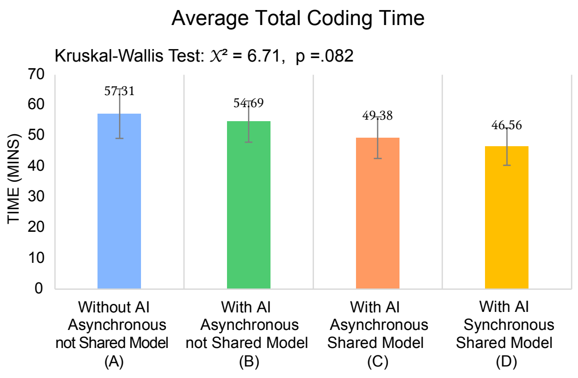


FIGURE 4.6: Average Total Coding Time for Each Condition (A, B, C, and D). Error bars show .95 confidence intervals. A Kruskal-Wallis test showed no main effect.

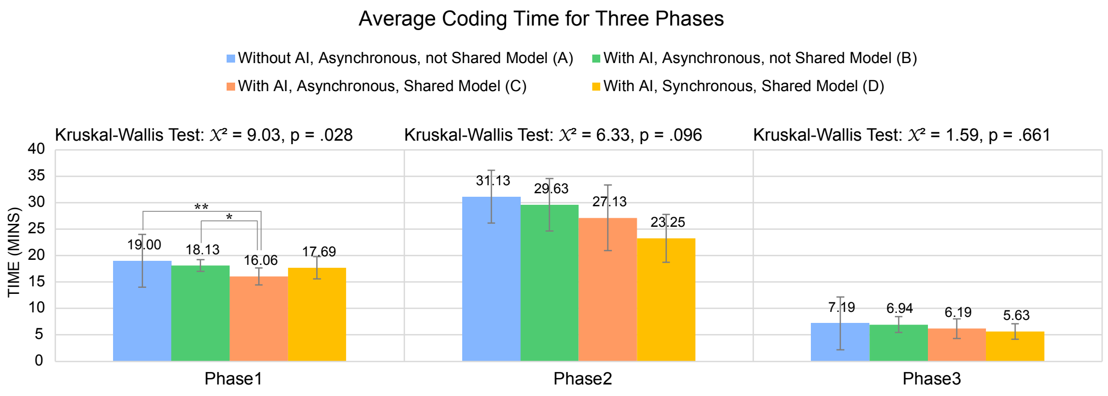


FIGURE 4.7: Average Coding Time for Three Phases. Error bars show .95 confidence intervals. We report the results of the individual Kruskal-Wallis tests, and, if necessary, pairwise comparisons, where $* : p < .05, ** : p < .01$.

coding time ($M = 16.06\ mins$) for *Condition C: With AI, Asynchronous, Shared Model* was significantly faster. In particular, we found a significant difference between *Condition C: With AI, Asynchronous, Shared Model* and *Condition A: Without AI, Asynchronous, not Shared Model* ($M = 19\ mins$, $U = 59.0$, $p = .005$). We also found a significant difference between *Condition C: With AI, Asynchronous, Shared Model* and *Condition B: With AI, Asynchronous, not Shared Model* ($M = 18.13\ mins, U = 51.5, p = .044$). No significant differences were found between *Condition D: With AI, Synchronous, Shared Model* and other conditions.

Overall, the average coding time for Phase 1 with AI conditions was decreased by 4.6%-15.5% compared to the baseline *Condition A: Without AI, Asynchronous, not Shared Model*. This also meant that AI conditions (B, C, D) resulted in a 9.0% faster coding time on average.

Phase 2

While we observed that *Condition D: With AI, Synchronous, Shared Model* was the fastest in the discussion phase, a Kruskal-Wallis test **did not show any significant main effect** of the condition on time. The time used for discussion in Phase 2 was between 23.25 mins for *Condition D: With AI, Synchronous, Shared Model* and 31.13 mins for *Condition A: Without AI, Asynchronous, not Shared Model* (see Figure 4.7).

Phase 3

Phase 3 was overall very fast, ranging from 5.63 mins for *Condition D: With AI, Synchronous, Shared Model* to 7.19 mins for *Condition A: Without AI, Asynchronous, not Shared Model*. We **did not find any significant main effect** of the conditions on time ($p = .661$).

4.5.2 Inter-rater Reliability

The average IRR ranges from 0.16 to 0.31 on average in Phase 1. The IRR then increased to 0.51-0.65 by the end of Phase 3, after discussion (Phase 2) (see Figure 4.8).

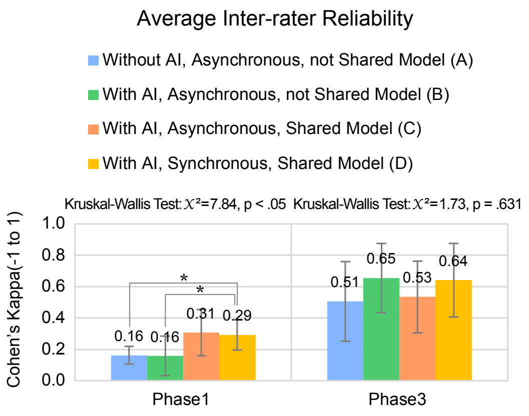


FIGURE 4.8: Average Inter-coder Reliability after Phase 1 and after Phase 3. Error bars show .95 confidence intervals. We report the results of the individual Kruskal-Wallis tests, and, if necessary, pairwise comparisons, where $* : p < .05, ** : p < .01$.

Phase 1

A Kruskal-Wallis test shows that there is **a significant main effect on the IRR** from the four conditions in Phase 1 ($\chi^2_{(3)} = 7.85, p = .049$). Post-hoc pairwise comparisons showed that IRR scores are a bit higher for AI conditions with Shared Model: IRR in *Condition A: Without AI, Asynchronous, not Shared Model* was significantly lower than *Condition D: With AI, Synchronous, Shared Model* ($U = 9.0, p = .015$). IRR in *Condition B:*

With AI, Asynchronous, not Shared Model was significantly lower compared to *Condition D: With AI, Synchronous, Shared Model* ($U = 13.0, p = .049$).

Phase 3

We did not observe **any main effect** between the four conditions for the IRR, ranging from 0.51 in *Condition A: Without AI, Asynchronous, not Shared Model* to 0.65 in *Condition B: With AI, Asynchronous, not Shared Model* ($p = .631$).

4.5.3 Code and Subcode Diversity

The terms, code (i.e., first-level code) and subcode (i.e., second-level code), are as defined as per section 4.4.6 and 4.4.6. The code and subcode diversity results are summarized in Figure 4.9. Our focus will be primarily on Phase 1 and Phase 2. We have opted not to include Phase 3 in our discussion, given that it employs the same codes and subcodes as Phase 2, thus having similar diversity.

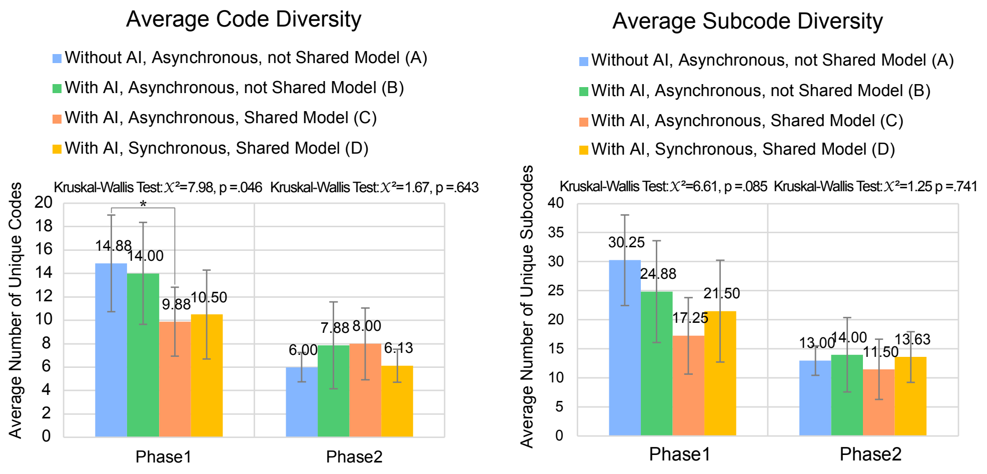


FIGURE 4.9: Average Code and Subcode Diversities in Phase 1 and Phase 2. Error bars show .95 confidence intervals. A Kruskal-Wallis test is conducted for the main effect in each phase. Pairwise comparison is performed using Mann–Whitney U-Test with a two-sided alternative, where $* : p < .05, ** : p < .01$.

Phase 1

A Kruskal-Wallis test **shows a significant main effect** of the condition on code diversity in Phase 1 ($\chi^2_{(3)} = 7.98$, $p = .046$). Pairwise comparisons show that the observed diversity was lower for conditions with AI and Shared model: the code diversity for *Condition C: With AI, Asynchronous, Shared Model* ($M = 9.88$ unique codes) was significantly lower than for *Condition A: Without AI, Asynchronous, not Shared Model* ($M = 14.88$, $U = 54.0$, $p = .023$).

Subcode diversity **did not seem to be impacted** by our four conditions in Phase 1 ($\chi^2_{(3)} = 6.61$, $p = .085$). However, it has an average number of unique subcodes ranging from 17.25 in *Condition C: With AI, Asynchronous, Shared Model* to 30.25 in *Condition A: Without AI, Asynchronous, not Shared Model*.

Phase 2

A Kruskal-Wallis test shows that **there was no main effect in terms of Code Diversity** after Phase 2 ($\chi^2_{(3)} = 1.67$, $p = .64$). The number of items decreased compared to Phase 1, with the average ranging from 6.00 (*Condition A: Without AI, Asynchronous, not Shared Model*) to 8.00 (*Condition C: With AI, Asynchronous, Shared Model*). **This lack of main effect was also visible for subcode diversity** ($\chi^2_{(3)} = 1.25$, $p = .74$) with the number of subcodes being between 11.5 items (*Condition C: With AI, Asynchronous, Shared Model*) and 14.00 items (*Condition B: With AI, Asynchronous, not Shared Model*).

4.5.4 Code and Subcode Coverage

We report and summarize the average code and subcode coverage in Figure 4.10.

Phase 1

We found that **no main effect for code coverage** between the four conditions in Phase 1 ($\chi^2_{(3)} = 4.79$, $p = .19$), where the coverage of code ranged from 0.75 (*Condition C: With AI, Asynchronous, Shared Model*) to 0.86 (*Condition A: Without AI, Asynchronous, not Shared Model*).

Phase 2

Similarly, **no main effect was found** after Phase 2 ($\chi^2_{(3)} = 1.78$, $p = .62$) with coverage ranging from 0.70 (*Condition C: With AI, Asynchronous, Shared Model*) to 0.80 (*Condition D: With AI, Synchronous, Shared Model*).

4.6 Triangulation with Qualitative Results

In summary, our quantitative results reveal nuanced disparities among our four conditions with respect to time duration, *IRR*, and code diversity in Phase 1. However, no significant differences were discerned within *Condition A: Without AI, Asynchronous, not Shared Model* and *Condition B: With AI, Asynchronous, not Shared Model*, **or** *Condition C: With AI, Asynchronous, Shared Model* and *Condition D: With AI, Synchronous, Shared Model*. Most

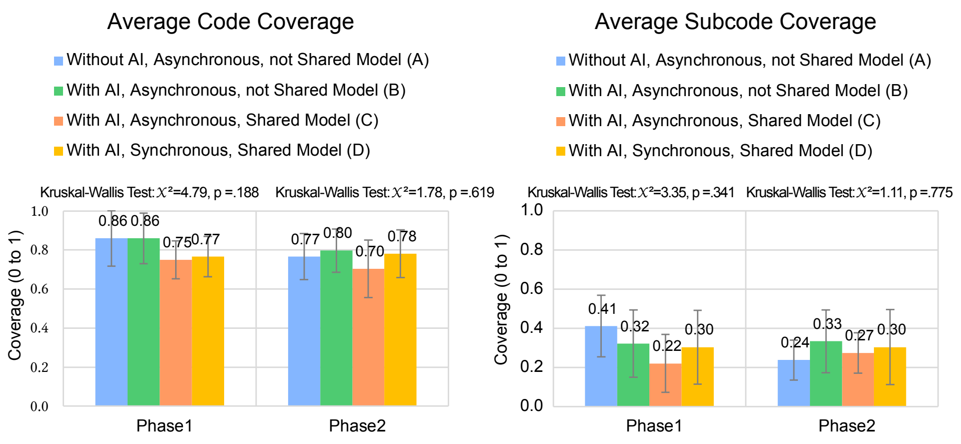


FIGURE 4.10: Average Coverage of Code and Subcode in Phase 1 and Phase 2. Error bars show .95 confidence intervals. We report the results of the individual Kruskal-Wallis tests.

observed variations, therefore, were between *Condition A: Without AI, Asynchronous, not Shared Model* and *Condition B: With AI, Asynchronous, not Shared Model* **on one side**, and *Condition C: With AI, Asynchronous, Shared Model* and *Condition D: With AI, Synchronous, Shared Model* **on the other**.

Despite the modest size of the statistically significant differences observed in our study, it is worth noting that even such small results can have meaningful implications (Altman and Bland, 1995; Hackshaw, 2008). The trends we discerned lead us to a two-fold set of primary findings: First, we found that in the context of our proposed CQA conditions, AI without a shared model may not improve coding efficiency as effectively as the shared model. Second, we found that combining AI with a shared model could potentially accelerate coding speed and achieve a higher level of initial *IRR*. However, this advantage came with a slight reduction in code diversity during the code development phase. To confirm our primary findings, we employ a triangulation method that combines qualitative results, as discussed in the following.

4.6.1 Lower Initial Coding Time

While our results indicate that only *Condition C: With AI, Asynchronous, Shared Model* significantly decreased the coding time in Phase 1, with *Condition D: With AI, Synchronous, Shared Model* not reaching statistical significance, there is a suggestive trend that participants under conditions with both AI and a shared model tend to engage in less discussion time in Phase 2 (refer to Figure 4.7) and less coding time overall (refer to Figure 4.6). The observed decrease in Shared Model conditions ranges from 13.8% to 18.9% for total time, 6.9% to 15.5% for Phase 1, and 12.8% to 25.3% for Phase 2. However, this trend does not extend to Phase 3. To validate these preliminary observations, a follow-up study with larger datasets would be beneficial in the future.

Additionally, it is important to note that due to study constraints, we had to enforce a time limit of 20 minutes for Phase 1, leading to the discontinuation of the coding task for four pairs who did not complete it within the regulated time (see section 4.4.6). Consequently, the differences observed between *Condition A: Without AI, Asynchronous, not Shared Model*, which closely adhered to the 20-minute limit in Phase 1 (refer to Figure 4.6), and the other conditions may actually be more substantial in real-world usage. However, further data would be required to support this conclusion.

4.6.2 Higher Initial IRR

Although we only detected a higher IRR for *Condition D: With AI, Synchronous, Shared Model* in comparison to *Condition A: Without AI, Asynchronous, not Shared Model* and *Condition B: With AI, Asynchronous, not Shared Model*, a significant upward trend emerged in the average IRR across Shared Model conditions, demonstrating an increase from 81.3% to 93.8% (refer to Figure 4.8).

This suggests the potential for a greater *IRR* in Phase 1 when utilizing AI & Shared Model, which could stem from the participants' capacity to utilize the Shared Model for reflection and code adjustments upon exposure to various code suggestions, primarily drawn from their counterparts. This early involvement in the "negotiation and merging" process enables participants to reach an agreement and shared understanding sooner in the initial coding stages.

Our observation is partially confirmed by the qualitative results. P33 from *Condition C: With AI, Asynchronous, Shared Model* noted, *"After I wrote (a code), I would check if the suggested codes are better."*. Similarly, in *Condition D: With AI, Synchronous, Shared Model*, P31 mentioned, *"It could have another word for 'introduction'. For example, maybe my partner will say 'intro'. But if you want to formalize things, then we realized that introduction is a more formalized code."*.

4.6.3 Lower Diversity

While the shared model enabled participants to save time and reach code convergence in the early phases of coding, it also resulted in lower code diversity. On average, in comparison to our baseline condition *Condition A: Without AI, Asynchronous, not Shared Model* (Mean = 14.88), the total number of unique codes decreased by 5.00 (in *Condition C: With AI, Asynchronous, Shared Model*) and 4.33 (in *Condition D: With AI, Synchronous, Shared Model*). This reduction can be attributed to the higher overall agreement among participants, which naturally limits the variety of codes used. Additionally, the usage of suggested codes, shared among participants in *Condition C: With AI, Asynchronous,* *Shared Model* and *Condition D: With AI, Synchronous, Shared Model*, further contributes to the decrease in code diversity.

4.6.4 Effect of Synchrony

It is interesting to note that the synchrony of coding (Synchronous or Asynchronous) did not seem to have an impact on CQA performance, as we did not observe any significant differences between *Condition C: With AI, Asynchronous, Shared Model* and *Condition D: With AI, Synchronous, Shared Model*. Nevertheless, we did observe effects from the qualitative results.

First, we noticed that some coders in *Condition C: With AI, Asynchronous, Shared Model* tended to rely more on the AI code suggestions, rather than proposing their own codes. This behavior might be attributed to their reliance on a model extensively trained using the first coder's codes. As a result, participants expressed concerns about potential bias introduced by the shared model and expressed worry that the code diversity and coverage in the open coding process (Phase 1) would be reduced. *"Bias was a bit problematic. Because when you're given suggestions that you might not need, but it shows it has a high confidence level, then subconsciously, I guess you might try to incorporate it [...] which you might not have done otherwise."* (P48 in *Condition C: With AI, Asynchronous, Shared Model*). This concern is substantiated by our quantitative results.

Second, differences in coding speed could contribute to a similar effect, as slower coders may end up reusing code generated by faster, synchronous coders. As one participant explained, *"We both generated a common knowledge base... but because I was doing slower, then a suggestion coming out with a 0.9 confidence score, which is the code I would have written. That's why I feel biased."* (P30 in *Condition D: With AI, Synchronous, Shared Model*).

It is crucial to note that the coders were not aware that the suggestions originated from the first coder's input, believing them to be AI-generated until it was revealed in the post-interview. We believe that leveraging the shared model is valuable; however, if the second coder was made aware of whether a suggestion originated from the AI or a previous coder, could alter their level of reliance on the suggestions and potentially impact the dependency between both coders.

4.6.5 Positive Feedback from Shared Model Conditions

While the use of a Shared AI model may involve a trade-off in terms of coding speed, *IRR*, and code diversity, it is notable that the conditions utilizing Shared AI models resulted in more positive experiences among participants. In both *Condition C: With AI, Asynchronous, Shared Model* and *Condition D: With AI, Synchronous, Shared Model*, 5 out of 8 pairs reported a smooth and swift coding process with the system.

Participants elaborated on the reasons behind their positive experiences. For instance, the shared model streamlined the coding process by enabling participants to reuse their own or their partners' previous codes. As one participant noted, *"Because it already has the options that I have entered before, it's faster if I want to add the same code in other paragraphs. I don't have to keep referring to what I wrote above."* (P37 in *Condition C: With AI, Asynchronous, Shared Model*). They expressed appreciation for the shared model's ability to offer timely suggestions, which substantially assisted them in refining their code expressions. This sentiment was aptly encapsulated by one participant who remarked, *"It sometimes has a better phrase or better word. You can just take from that."* (P34 in *Condition D: With AI, Synchronous, Shared Model*).

The shared model also facilitated coders in aligning their understanding with their partner's during the coding process. As one participant explained, *"For me, yes, coding efficiency was improved. Because my partner was coding just the main points... When I was doing my own coding, I could see [these] main points. It definitely helped me understand what was going on."* (P17 in *Condition D: With AI, Synchronous, Shared Model*). This can also prompt them to compare their own viewpoints against a broader range of perspectives. P48 in *Condition C: With AI, Asynchronous, Shared Model* noted, *"So [AI] suggested `weakness` [as a code], I thought it could be `society`. I want to say this kind of inconsistency is just a difference of opinion [of what] terms or labels here."*

In contrast, the majority of participants expressed a neutral or negative attitude towards coding efficiency, while only one participant (P63) from *Condition B: With AI, Asynchronous, not Shared Model* perceived an improvement.

4.6.6 Similarity of Codebooks across Conditions

Overall, our main DVs allow us to understand how AI might help with coding. However, it can still be hard to find out whether the results across our four conditions are similar or at least comparable. We thus identify the 7 most common first-level codes in the formed codebook for each condition (see Appendix Table A.1). Overall, the themes are rather similar (despite a slightly different ranking), suggesting that the results are consistent across the four conditions. Most code categories appeared in all four codebooks, e.g. introduction, leadership, weakness, hiring.

4.7 Discussion

4.7.1 Trade-off: Coding Efficiency vs. Coding Quality

Overall, the introduction of AI with Shared Model has the potential to streamline the early coding process, saving time and fostering early consensus on codes, as reflected by higher initial IRR. However, this comes at the cost of reduced initial code diversity.

We recognize the inherent trade-off between (Coding Time & IRR) and Diversity — essentially, the balance between **coding efficiency and coding quality** — as a persistent characteristic of the early stages of CQA, particularly when seeking shared AI mediator assistance.

AI & Shared Model Fosters Strong Discussions

Previously, we referenced Zade et al., who identified two types of disagreements that can occur during coding: 'diversity' (varying interpretations of a single core idea) and 'divergence' (distinct core ideas) (Zade et al., 2018).

For the former, we found users often favor consistency in coding outcomes across coders over excessive diversity. In this context, the shared model proves beneficial, aiding coders in refining their phrasing. For the latter, we noted that while coders appreciated AI code suggestions—rooted in their collective coding history—they still actively formulated their own codes, especially when facing divergent disagreements.

This process encourages real-time comparison and validation, prompting coders to critically reassess their coding decisions against alternate perspectives. When incorporated into subsequent discussions, these reflective insights enrich the overall dialogue. As a result, AI can serve a crucial role as a facilitator in human-to-human collaboration, stimulating more substantive and engaging discussions.

Potential Pitfalls

While AI and Shared Model bring considerable benefits, it is crucial not to overlook the following two potential challenges.

Reduced Diversity. We argue that code diversity, by introducing varied perspectives, plays a crucial role in enhancing coding quality within the CQA process (Richards and Hemphill, 2018; Anderson, Guerreiro, and J. Smith, 2016). It is crucial to ensure that coding practices do not excessively compromise coding quality or diversity. Otherwise, users may resort to traditional tools that ensure coding quality through extensive discussions between collaborators on a line-by-line and code-by-code basis. Therefore, we call for the need for further research in this area to address this challenge. For instance, one approach is to enable the system to generate synonyms or keywords of potential codes as references for coders, apart from the original code suggestions (Marathe and Toyama, 2018b). Moreover, allowing the system to easily highlight nuanced differences and compare text selections, by linking back to prior data with simple clicks, can deepen coders' understanding and thus yield more insightful codes.

Over-reliance. Moreover, we must acknowledge the potential risk of users becoming overly dependent on the system. This tendency can be exacerbated when users aim to increase their speed, encounter challenges in formulating code names, or face discrepancies in coding speed among coders. Such over-reliance could potentially decrease the propensity to seek different opinions, leading users to only choose from system suggestions, particularly during asynchronous coding. This also raises concerns about a potential loss of diversity and nuance in coding, especially in loosely-defined coding tasks where multiple iterations may be necessary to reach consensus on a codebook. In future studies, measures to curb over-reliance on the AI system could include providing explanations for code suggestions (Vasconcelos et al., 2023), interface warnings to indicate excessive system use, or even temporary system disabling if over-reliance is detected.

4.7.2 Is Shared Model best for CQA? Considering Different Application Scenarios

In light of the points discussed in the previous subsections, we do not argue that AI & Shared Model is the best solution for CQA.

Supporting Different Contexts with Different Independence Level.

Our evaluation revealed that the four collaboration methods effectively simulated four distinct levels of independence, each potentially desirable in different real-life scenarios. In conditions without AI or Shared Model (Condition A, B), coders maintained the highest level of independence, with no communication during the code development phase. Conversely, under conditions with AI & Shared Model (Condition C, D), coders demonstrated a lower level of independence—indirectly connected and communicating through the AI mediator.

The evaluation of different levels of independence yielded varying results. A higher level of independence was found to generate greater code diversity with lower efficiency, while a lower level of independence resulted in lower code diversity with higher code efficiency. Align with this, the feedback obtained from participants during the interview (see section 4.2.2) indicates that they usually compromise some independence in favor of enhanced coding efficiency when time constraints are present.

The observation, coupled with our formative interviews, has enabled us to discern two distinct scenarios in the context of CQA: efficiency-oriented and creativity-oriented.

In the former, researchers facing time constraints or requiring both qualitative and quantitative results may be inclined to sacrifice some rigidity in coding for improved coding efficiency. As a result, they might prefer alternative methods, such as using pre-defined themes in analysis, over strict adherence to the Grounded Theory process, which is often seen as more inductive, requiring meticulous, line-by-line open coding

(Maguire and Delahunt, 2017). Therefore, when integrating AI to facilitate human-to-human collaboration, some level of communication between coders may be beneficial and acceptable. However, efficiency should not be prioritized to the point of jeopardizing code diversity or undermining the primary goal of CQA—to glean multiple perspectives.

For the latter, in research domains that might heavily rely on creativity, perspective, and critical thinking (e.g., Psychology, Anthropology), it is crucial to minimize external influences and maintain a high degree of independent thought and self-autonomy. For instance, researchers may choose to use separate coding models that generate code suggestions based on each coder's individual coding history.

In summary, we underscore the need to **consider varying levels of independence across diverse contexts**, as this can impact the trade-off and balance among different coding outcomes when leveraging an AI mediator to facilitate human-to-human collaboration within the realm of CQA.

Support Different User Groups with AI & Shared Model

We believe that CoAIcoder has the potential to be beneficial for various user groups. In our study, we involved participants who had limited experience with qualitative coding. Based on the feedback from interviewees, this particular user group frequently engages in CQA analysis and codebook creation (see section 4.2.2).

CoAIcoder for the Learning purpose of Novices During our study, participants expressed a positive reception towards the code suggestions, particularly in the initial stages of the coding process. One participant mentioned, *"Firstly, I did not know what to put because I don't know what is required"* (P49 in *Condition C: With AI, Asynchronous, Shared Model*). Another participant stated, *"Quite useful, because at least I can start. That's how I'm supposed to do it"* (P36 in *Condition C: With AI, Asynchronous, Shared Model*). These responses indicate that participants found the system helpful in overcoming the initial challenges of understanding coding requirements and getting started with the analysis.

Moreover, participants reported that the system was easy to use. As one participant remarked, *"It seems pretty easy to use"* (P63). By providing a user-friendly interface, our system addresses the learning curve issues highlighted in prior research (J. A. Jiang et al., 2021; Yan, McCracken, and Crowston, 2014) associated with conventional qualitative QA software like nViVo and Atlas.ti. This suggests potential to mitigate users' reliance on traditional collaboration software like Google Docs and Sheets, which are not specifically designed for qualitative analysis.

CoAIcoder for Expert Users We contend that expert coders might derive benefits from our system, as it has the potential to save their time and facilitate higher levels of agreement through AI mediation. We also speculate that expert users may not rely on the system heavily, thereby mitigating the risk of over-reliance. However, further evaluation with expert users is necessary to substantiate this hypothesis.

4.8 Design Implications

Beyond insights into human-to-human collaboration via AI mediation, we also identify design implications for human-AI collaboration, such as the impact of coding granularity and the cultivation of trust between humans and AI.

4.8.1 Impact of Coding Granularity on Human-AI collaboration

Coding granularity includes the unit-of-analysis (UoA) and code specificity. The UoA delineates the level at which text annotations are made, for example, on a flexible or sentence level (Rietz and Maedche, 2021). Code specificity refers to the varying degrees of detail in a code, for instance, it can range from a broad code to a more specific one. In our evaluation, we did not regulate the UoA and code specificity and left the selection of text and codes open. This approach mirrors real-world coding processes, where individuals often apply codes at various units and code specificity (Rietz and Maedche, 2021). We noted two significant implications arising from this setup.

Establishing Optimal Coding Granularity for both AI and Human Coders

We first observed a significant variation in coders' interpretation and application of codes, from broad generalities, which often lack informational depth, to more specific interpretations that may not apply universally. A participant clarified, *"So let's say for 'leadership' code, right? We should write like 'introduction to leadership' and then it can be applied across the 200 (transcripts). If you write like 'introduction to event planning', we can't use it, as not everybody organized [event]."* (P17 in *Condition D: With AI, Synchronous, Shared Model*).

AI generates suggestions by learning patterns and structures in the data it has been trained on. They are more likely to generalize based on the most prominent features or patterns in the data instead of understanding context, nuance, and the complexity of human language and emotions, as noted by a participant in Jiang et al.'s study (J. A. Jiang et al., 2021). This limitation can have significant implications for how coders approach their work. If coders become aware that AI systems are more likely to suggest or recognize broad codes, they might deliberately write broader codes to ensure compatibility with AI suggestions, which could lead to oversimplification of the data and possibly missing out on nuanced or intricate details. It also highlights the necessity for human involvement in AI-assisted qualitative coding. The human coder's role becomes critical in guiding the AI, providing the nuance, context, and deeper understanding that the AI may lack. Strategic considerations include the level of specificity in code selection and the possibility of optimizing codes that can be efficiently processed by both AI and human understanding. The goal is to design an AI-assisted system that is not completely dependent on AI but instead integrates the strengths of both AI and human coders to overcome the identified limitations.

Impact of Coding Granularity on IRR Calculation

Another challenge we encountered was the evaluation of quality. This issue becomes particularly problematic when calculating IRR, as precise numeric codes on similar levels of the text are usually required for its calculations. When coders choose codes or text of varying levels, it can lead to ambiguity (Ceccato et al., 2004) as it becomes challenging to determine if they've assigned identical codes to a particular unit. To address this challenge, a potential strategy is to regulate coding units or pre-discuss "soft rules" for agreed-upon levels of units (e.g., sentences, paragraphs) before conducting coding (Kurasaki, 2000; O'Connor and Joffe, 2020). However, to maintain user flexibility in our study, we opted to map the assigned codes to sentence level after coding for IRR calculation. Future research can explore the combination of these two methods to maximize the advantages they bring.

Impact of Coding Granularity on Stability of Suggestions

We noted occasional system instability due to frequent retraining, which could change suggestion order or composition and disrupt established interaction patterns. One participant noted, *"When I selected one sentence of this paragraph, the code is there. But when I select the whole paragraph, the [same] code is not there [any more]."* (P47 in *Condition C: With AI, Asynchronous, Shared Model*). This instability presents challenges for coders' user experience who predictively interact with the system and rely on prediction stability (H. Liu et al., 2022). However, users primarily using the system for decision-making aid and consistently choosing from the suggestion list might have minimal reliance on prediction stability. For them, minor alterations might not pose significant issues, as long as the system continues to provide relevant and accurate suggestions. Furthermore, a user's dependence on prediction stability could also evolve with familiarity with the system. Novice users might rely heavily on the system's suggestions stability, while experienced users might develop their own interaction strategies. From the model's perspective, prediction stability depends on the extent of data changes between training iterations. If the newly added data closely mirrors the pre-existing data, the model's predictions could remain stable. Conversely, if the new data significantly diverges, predictions might undergo noticeable changes. Employing strategies like incremental learning, where the model learns from new data without forgetting previous knowledge, could potentially maintain stable predictions (Giraud-Carrier, 2000).

4.8.2 Trust and User Expectations

While we strive for maximum accuracy in AI systems, it is important to acknowledge that achieving perfection, especially in tasks involving subjective data, is challenging (N.-C. Chen et al., 2018; J. A. Jiang et al., 2021).

Calibrating Users' Expectation Before Coding

Previous research has demonstrated that unrealistic user expectations can lead to reduced user satisfaction with AI systems (Kocielnik, Amershi, and Bennett, 2019; Grimes, Schuetzler, and Giboney, 2021; Ashktorab, Liao, et al., 2020; Cheng et al., 2019). In our evaluation, participants exhibited a notable initial expectation regarding the AI's capability to provide suggestions. If the system failed to meet their expectations, participants resorted to manually entering their own codes. Additionally, the occasional inaccuracies in the AI's suggestions brought confusion and might damage their confidence in the AI. *"For one sentence I thought it was "interest". The number [for the other suggestion] was like 0.9 but for "interest" was like 0.00 something. I was a bit confused."* (P62 in *Condition B: With AI, Asynchronous, not Shared Model*).

Additionally, participants expressed a desire for more timely results when requesting code suggestions. However, our current "retrieve-train-predict" process takes at least 10 more seconds, even when utilizing a GPU. This delay in providing suggestions may have impacted participants' willingness to utilize the AI suggestions. *"One difficulty is the code I put in takes a while to come out when I want to use it again for another passage."* (P38 in *Condition C: With AI, Asynchronous, Shared Model*).

In future work, in addition to enhancing the accuracy and stability of the system, managing users' expectations will be crucial. One approach could involve calibrating users' expectations by providing information about the system's capabilities, expected accuracy, and the possibility of errors. Furthermore, making the suggestions more explainable and interpretable could provide users with insights into the underlying reasoning, potentially easing users' doubts, distrust, and frustration (Knowles et al., 2015; Golafshani, 2003; Lubars and Tan, 2019; Liao, Gruen, and S. Miller, 2020).

Can Imperfect Suggestions Help?

Research suggests that even imperfect AI can still provide valuable assistance to users (Kocielnik, Amershi, and Bennett, 2019). In our case, when AI suggestions are in conflict with the assumptions made by coders, it indicates the possibility of either partially incorrect or completely incorrect suggestions. When AI suggestions partially align with users' thoughts, they can select the suggested codes and implement minor adjustments. Therefore, these suggestions can still assist users in making code decisions, and this valuable user input can contribute to improving the model's performance in subsequent training. When AI suggestions are completely not matched, users have the option to bypass the suggestions and manually enter their own codes.

However, some coders may not be aware of the aforementioned strategy. When they encounter imperfect suggestions, their trust in the AI system can diminish. *"I guess I did not use [the AI suggestions] much, because the words I needed wouldn't not be suggested. But I guess if one suggestion is a bit close to what you are thinking, it's enough. But if you want to put your exact thoughts, then I guess doing it manually would be better."* (P48 in *Condition C: With AI, Asynchronous, Shared Model*). Sometimes, a relevant code suggestion may not be among the top five suggestions but could instead appear within the top ten. Therefore, it would be beneficial for users to have access to a longer list of codes upon request. While this feature may not be frequently utilized, it would grant users more control over the system. *"I think for myself, [the system] wasn't that helpful... because there's only five in the drop-down list. Even if I want something that is exactly the same words, but it's not in the top five recommended, I cannot get it. There isn't a scroll down or something."* (P32 in *Condition C: With AI, Asynchronous, Shared Model*).

Overall, it is important to communicate to coders about the capabilities of the AI system and how to effectively respond to imperfect code suggestions. Such information can enhance coders' understanding of AI system's capabilities and limitations, allowing them to fully leverage it.

4.9 Limitations and Future Work

This work has limitations. First, our research primarily involved novices participating in the system evaluation. There were several reasons for this choice: 1) based on our initial interviews, training novice users and integrating them into the CQA team is a common practice (see section 4.2.2); 2) in our investigation of collaboration factors within the CQA context (i.e., With/Without AI, Synchrony, Shared Model/No Shared Model), designing a between-subject design was necessary. Ensuring a similar baseline experience among participants was also essential to maintain fairness across conditions. In the study, we took measures to promote accurate task completion: a) we provided thorough CQA training to all participants in each session, aiming to equip them with the skills necessary to efficiently carry out the tasks; b) we monitored the participants' coding results and the quality of their work closely during the task to minimize disruption to their coding process (see section 4.4.4); c) after collecting data, we carried out quality checks to verify the reliability and completeness of the results (see section 4.4.6).

However, despite these measures, it's important to acknowledge that our findings might differ from those obtained from seasoned CQA experts. For example, in some isolated cases, we observed a slight over-reliance on the system by novices. However, even when employing novices, the need for discussion and collaboration persists. Therefore, experts could also derive benefits from the time-saving capabilities of our system. Future research should broaden the user base to include expert-expert and expert-novice pairs, as well as native-nonnative speakers, to explore other potential advantages of the various collaboration diagrams.

Another limitation of our study is the lack of substantial statistical significance, which can potentially be traced back to multiple factors. One key aspect is the inherent complexity of human collaboration in the context of CQA tasks, which is multifaceted and nuanced, carries considerable significance. For example, the diversity of reactions among coders to AI-suggested codes, or the differences in the pace at which individual coders learn and adapt to the coding process. Additionally, coders might have widely different coding strategies that influence their coding speed and the quality of their output. Furthermore, while we opted for a potentially optimal number of AI suggestions for the CoAIcoder's design in this iteration, the influence of the number of suggestions on user behaviors is an important design aspect to consider. It could also be interesting to examine the impact of revealing AI suggestions either before or after users select the text.

Another consideration lies in system-related factors, including the frequency of training updates and delays in achieving stable model accuracy, which are certainly significant. It is worth noting that improving model performance on subjective annotation continues to be a key challenge in fields like NLP (Davani, M. Díaz, and Prabhakaran, 2022) and HCI (N.-C. Chen et al., 2018), and this aspect remains under-explored in the context of CQA. In future work, it would be advantageous to incorporate data augmentation technologies or other NLP pipelines to enhance model performance. Moreover, the promising potential of advanced LLMs, e.g., GPT-4[9], which demonstrate exceptional capabilities in understanding and generating text (Z. Zhang et al., 2023), should not be overlooked. They could be harnessed to facilitate AI-assisted qualitative coding[10] as well as CQA (Gao, Guo, Lim, et al., 2023).

These individual differences and system-related factors, though subtle, can cumulatively exert a profound impact on collaborative coding dynamics and the detection of the final significant difference. However, they remain largely unexplored. While we recognize their significance, our main emphasis in this work lies in unearthing the possibilities and influence of AI mediation on human collaborative dynamics within a CQA context. We strongly recommend further research in this field, given the significant role of CQA

[9] `https://openai.com/research/gpt-4`

[10] Announced on 28th March 2023: `https://atlasti.com/ai-coding-powered-by-openai`

in qualitative research and the current focus primarily on AI's application in individual qualitative analysis. Our goal is to pinpoint and investigate the research gap, rather than establish a definitive or arbitrary methodology for AI-assisted CQA.

4.10 Conclusion

In this work, we delve into AI-assisted human-to-human collaboration within the context of CQA and assess various collaboration modes between coders. To the best of our knowledge, this marks the first attempt to investigate the role of AI in the collaboration of qualitative coding, as previous research primarily concentrated on individual coding with AI. In pursuit of this goal, we initially gained insights into coders' CQA behaviors, challenges, and potential opportunities through a series of semi-structured interviews. Following this, we developed and implemented a prototype, CoAIcoder, designed to provide coders with code suggestions based on their coding history. We further introduced four collaboration methodologies and evaluated them using a between-subject design with 64 participants (32 pairs). This led to the identification of the trade-off between coding efficiency and coding quality, as well as the relationship between independence level and the coding outcomes under varying CQA scenarios. We also highlighted design implications to inspire future CQA system designs.

Chapter 5

Investigating the Impact of Human-AI Interaction on User Trust and Reliance in AI-Assisted Qualitative Coding

5.1 Motivation

While substantial research exists in this field, a comprehensive examination of user reliance, trustworthiness and helpfulness of AI-assisted systems remains scarce (J. A. Jiang et al., 2021). To our knowledge, this work represents the first attempt to deeply explore how different human-AI interaction strategies within qualitative analysis impact the user trust and reliance for *AIQCs*.

User trust and reliance are fundamental factors to consider in constructing human-centric AI systems (Bach et al., 2022; Vereschak, Bailly, and Caramiaux, 2021). Lee et al. (J. D. Lee and See, 2004) distinguish "trust" as an "attitude" and "reliance" as a specific "behavior". Such attitudes can shape intentions, which subsequently manifest as behaviors. While trust is challenging to quantify directly, behavioral metrics like reliance—evident in how often users accept or follow recommendations—can act as indirect indicators of trust (Papenmeier, Kern, Englebienne, et al., 2022). Within the *AIQCs* context, trust issues have also been identified. Jiang et al. (J. A. Jiang et al., 2021) highlight the numerous factors that contribute to distrust between humans and AI. These include user's skepticism towards the AI's capability to execute qualitative analysis reliably, noticeable behavioral disparities between humans and AI, the absence of explanations of AI suggestions, and so on.

Based on their work, we argue that varying human-AI interactions in qualitative analysis, which arise from different coding strategies—a factor that has seemingly been overlooked—pose unique challenges for users to build appropriate trust and reliance on AI throughout the *Open Coding* process. Depending on the final objective, coding approaches can vary greatly. For instance, some might opt to code entire paragraphs with a concise code, providing a broad classification for large text segments. Conversely, others might focus on phrases, applying lengthier codes for more detailed insights. Despite similar processes, these strategies yield vastly different depths and scopes in coding outcomes.

In particular, there is a cascading chain of influence—from the interaction between humans and AI to the user trust and reliance: the ① **human-AI interaction** can substantially vary based on the ② **coding strategies** employed. It influences the ③ **users' input**—essentially, the ④ **training data for the AI**—which consequently impacts both the ⑤ **model's performance** and, therefore, the ⑥ **quality of AI suggestions**. This chain of influences, in turn, shapes ⑦ **humans' trust and reliance** in AI systems. Consequently, our goal is to investigate the impact of coding strategy and its associated influence chain on the dynamics of user trust and reliance. This exploration aims to uncover enhanced design insights for *AIQCs*.

5.2 Background and Related Work

5.2.1 Trust and Reliance with *AIQCs*

Trust encompasses a range of definitions in the literature (Vereschak, Bailly, and Caramiaux, 2021; Dzindolet et al., 2003; Mayer, Davis, and Schoorman, 1995; J. D. Lee and See, 2004). Lee (J. D. Lee and See, 2004) differentiates between "reliance" as a behavior and "trust" as an attitude. An attitude serves as an emotional assessment of beliefs that inform one's intentions, which subsequently manifest in behavior. Reliance, characterized by how frequently a user depends on or concurs with a system, serves as a behavioral metric to assess trust and evaluate user attitudes towards system usage (Papenmeier, Kern, Englebienne, et al., 2022; Dzindolet et al., 2003).

As trust and reliance have become a significant topic in the fields of CSCW and HCI (BANOVIC et al., 2023; Bach et al., 2022; Vereschak, Bailly, and Caramiaux, 2021), there is also a growing interest in exploring and establishing this element within the *AIQCs* domain. In the interview conducted by Jiang et al. (J. A. Jiang et al., 2021), the authors highlight several sources that contribute to distrust in AI. For instance, they point out discrepancies in the "typical behavior of humans and AI", as AI offers direct suggestions even when humans cannot provide a specific "correct" recommendation, and AI tends to prioritize suggestions with higher probabilities rather than subtle and nuanced insights. They also pointed out that low-precision models often require extra human effort for corrections. Moreover, the absence of explanations from AI and skepticism regarding AI's capacity for creativity and serendipity frequently lead to increased distrust. On the contrary, excessive reliance on AI might prompt researchers to defer to AI as the ultimate authority, potentially compromising human deliberation in the decision-making process.

Expanding on Jiang et al.'s work, we delve into trust nuances, focusing particularly on the reliance aspect between humans and AI. We mainly assess objective behavioral measures, such as the extent to which users accept code suggestions. In the context of *AIQCs*, it's crucial for users to strike the right balance in reliance—avoiding both under- and over-reliance. Moreover, by emphasizing reliance, we aim to gain insights into user interactions with AI systems perceived as less trustworthy or flawed, especially in subjective tasks (BANOVIC et al., 2023; Kocielnik, Amershi, and Bennett, 2019; N.-C. Chen et al., 2018; N.-c. Chen et al., 2016).

This is supplemented by subjective evaluations, like self-reported scores, to gauge users' trust or users' perceived trustworthiness in a system. Trustworthiness here is defined as "the extent to which the trustee believes that an automated system will behave as expected" (Papenmeier, Kern, Englebienne, et al., 2022; Vereschak, Bailly, and Caramiaux, 2021). We emphasize user perception since, in qualitative analysis, participants typically perceive the AI's capability to "reliably provide suggestions" as low (J. A. Jiang et al., 2021). In addition, we also measure users' *Perceived Helpfulness*, is because it is described as "the extent to which users perceive the recommendation as being capable of facilitating judgment or decisions" (Qin and Kong, 2015; M. Li et al., 2013). There is a strong connection between *Perceived Trustworthiness* and *Perceived Helpfulness*: if users find recommendations helpful, they are more likely to seek advice from those recommendations, thereby fostering trust in the system's capabilities. By considering both behavior and perception, we aim to identify potential discrepancies and foster deeper insight into users' holistic trust in the system (Scharowski et al., 2022; S. Cao and Huang, 2022; Brachman et al., 2022).

While current *AIQCs* approaches typically rely on human input for model training and generating code suggestions, there is a lack of research examining the human-AI interactions within the context of *AIQCs*. These interactions may introduce unique challenges specific to *AIQCs* that can significantly influence user trust and reliance. Certain interactions have the potential to generate higher-quality input, leading to more accurate code suggestions and improved assistance, while others may have a detrimental effect on these outcomes, subsequently shaping user perceptions and ultimate reliance on the systems.

5.2.2 Human-AI Interaction within AIQCs

There has been significant interest in finding ways for end-users, rather than experts, to interact meaningfully with machine learning systems in order to enhance system performance and user experience (Stumpf et al., 2009; Amershi et al., 2014). Researchers have also investigated how to refine features of machine learning systems by incorporating human perspectives, particularly in complex qualitative content analysis scenarios (Liew et al., 2014).

For *AIQCs*, AI often relies on human-generated training data, which serves as model input. However, unlike many traditional AI tasks, the inherently subjective nature of qualitative analysis poses a unique challenge. This challenge stems from the difficulty of obtaining specific and consistent human inputs, such as labels and text, for AI models. A major contributing factor to this issue is the variability in code granularity present in human coding (Saldaña, 2021; Lindgren, Lundman, and Graneheim, 2020). This variability may lead to inconsistencies and unreliability in the produced data, subsequently affecting the performance and trustworthiness of *AIQCs*.

Text Granularity denotes the specific selections of text that are to be coded. Imagine an *Open Coding* exercise where coding occurs on a word-by-word basis. In such a scenario, the overall context of the text is lacking, making it difficult for the model to suggest any useful codes. Consequently, coders may lose trust in the AI's ability and decide to stop using the system. On the other hand, performing line-by-line or sentence-by-sentence coding (Saldaña, 2021) would provide the AI with more context. This increased context could potentially enhance the performance of AI models (Farra et al., 2010), thereby influencing the system's perceived trustworthiness among users. In this work, we have chosen to examine three different levels of text granularity: **sentence, paragraph, and selective**. For the last level, users can select phrases of any length, which more closely resembles a regular *Open Coding* process.

Code Granularity refers to the length and specificity of a code (Lindgren, Lundman, and Graneheim, 2020). When a code is short, broad, and general (e.g., "experience", "leadership"), the AI might exhibit commendable performance from a classification perspective: the probability of AI suggestions aligning with the user-selected text is elevated, thereby expanding the pool of potential choices within the AI's suggestions. Despite this, dependency on AI assistance under these circumstances is not advisable. There's a risk of users becoming overly dependent on it, which may undermine the depth and diversity of qualitative analysis. On the other side, if users add excessively lengthy or detailed codes (e.g., "he hosts lots of activities", "her pets are very cute"), they may not serve as suitable categories for classification as they could exhibit limited commonality for code reuse. This situation could potentially impact the performance of the AI models. In this work, we have elected to explore three distinct levels of code granularity: **Short Codes** (i.e., Concise and General Codes), **Long Codes** (i.e., Detailed and Comprehensive Codes), and **Mixed Codes** (Natural Codes). In the case of the latter, users may employ code lengths ranging from one to six words, more closely mirroring a typical *Open Coding* process.

Both of the aforementioned scenarios could hinder the AI model from performing as well as expected. As a result, AI could become less useful, leading users to either under-utilize it or rely on it excessively. Hence, the granularity levels for both codes and text selections must be carefully designed, to enhance the clarity and consistency of the coding scheme (N.-c. Chen et al., 2016; Marathe and Toyama, 2018b; Rietz and Maedche, 2021). We anticipate that our evaluation will distill key insights, thus aiding in the creation of trustworthy, reliable, and beneficial *AIQCs*.

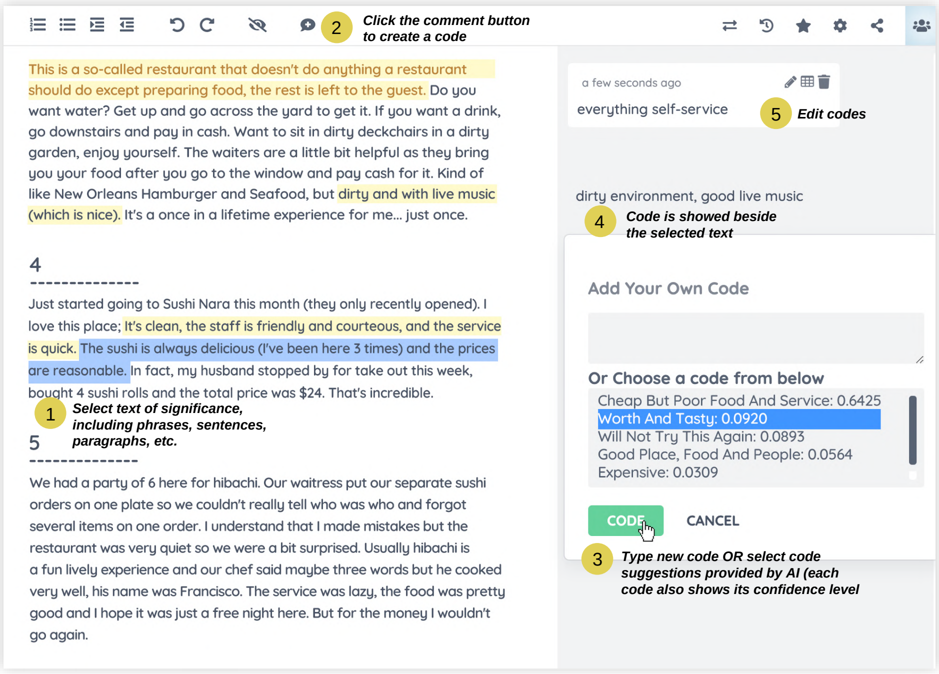
5.3 AIcoder: AI-assisted Qualitative Coding Tool

We used the prototype in CoAIcoder work, everything is the same except that at this time we only want to explore the interaction between human and AI instead of two humans and AI. So we only used one coder's codes as training data to fine-tune the pre-trained model. The interface of *AIcoder* is displayed in Figure 5.1, while the backend structure is depicted in Figure 5.2. With *AIcoder*, users can conveniently highlight any segment of text and assign a specific code. Following the coding, the selected text snippets and their corresponding codes are utilized to fine-tune a model based on their inputs. Ultimately, the user can highlight another piece of text to receive several recommendations from the model.

5.4 Study Design

We conducted a user study to assess the impact of various coding strategies on user trust and reliance in *AIcoder*. To establish different levels of granularity described in section 5.2.2, we set parameters for the length of the text selection and the associated code. Users were then asked to undertake qualitative coding at the sentence level, paragraph level, or with more flexible selection within a paragraph. Additionally, they were requested to summarize their codes in either a concise manner (short phrases of no more than three words), a more extended format (phrases containing four to six words), or in a freer style (phrases of mixed lengths ranging from one to six words). These specifications were derived from pilot studies conducted prior to the formal study.

Ultimately, we evaluated the model's performance (RQ1 in section 5.4.7); the *Decision Time* and *Coding Behavior* (RQ2 in section 5.4.8); the *Selecting Rate* and user reliance examination (RQ3 in section 5.4.9); participants' self-reported trust in *Perceived Trustworthiness* and *Perceived Helpfulness* of the system (RQ4 in section 5.4.10); as well as their subjective preferences (RQ5 in section 5.4.11).


2 Click the comment button to create a code
a few seconds ago
everything self-service
5 Edit codes
dirty environment, good live music
4 Code is showed beside the selected text
Add Your Own Code
Or Choose a code from below
Cheap But Poor Food And Service: 0.6425
Worth And Tasty: 0.0920
Will Not Try This Again: 0.0893
Good Place, Food And People: 0.0564
Expensive: 0.0309
CODE
CANCEL
3 Type new code OR select code suggestions provided by AI (each code also shows its confidence level
1 Select text of significance, including phrases, sentences, paragraphs, etc.

This is a so-called restaurant that doesn't do anything a restaurant should do except preparing food, the rest is left to the guest. Do you want water? Get up and go across the yard to get it. If you want a drink, go downstairs and pay in cash. Want to sit in dirty deckchairs in a dirty garden, enjoy yourself. The waiters are a little bit helpful as they bring you your food after you go to the window and pay cash for it. Kind of like New Orleans Hamburger and Seafood, but dirty and with live music (which is nice). It's a once in a lifetime experience for me... just once.

4

Just started going to Sushi Nara this month (they only recently opened). I love this place; It's clean, the staff is friendly and courteous, and the service is quick. The sushi is always delicious (I've been here 3 times) and the prices are reasonable. In fact, my husband stopped by for take out this week, bought 4 sushi rolls and the total price was $24. That's incredible.

5

We had a party of 6 here for hibachi. Our waitress put our separate sushi orders on one plate so we couldn't really tell who was who and forgot several items on one order. I understand that I made mistakes but the restaurant was very quiet so we were a bit surprised. Usually hibachi is a fun lively experience and our chef said maybe three words but he cooked very well, his name was Francisco. The service was lazy, the food was pretty good and I hope it was just a free night here. But for the money I wouldn't go again.

FIGURE 5.1: *AIcoder* Interface. The above figure shows a user was doing coding using *Mixed Codes*. The user can add codes by 1) selecting the text of significance or interest, including phrases, sentences or paragraphs, etc.; 2) clicking the comment button to create a code; 3) typing new code or selecting code suggestions suggested by AI. Each code also shows the confidence level, ranging between 0 and 1; 4) code is shown beside the selected text; 5) edit codes.

5.4.1 Study Task

Dataset

We selected the reviews at random from the publicly accessible Yelp reviews dataset[1] for our open coding task. We chose Yelp reviews due to two main reasons. First, the content of reviews is a form of text that most people are familiar with, thus facilitating the coding process for participants without imposing significant difficulties. Second, reviews often come in short paragraphs, increasing the likelihood that a code assigned to one paragraph could also apply to another.

Pilot Test

In order to ascertain that our participants could complete the open coding tasks within a reasonable timeframe of 1 to 1.5 hours, we conducted a pilot study involving 6 graduate students with proficient English skills. This exercise revealed that a coding

[1]https://www.yelp.com/dataset/documentation/main

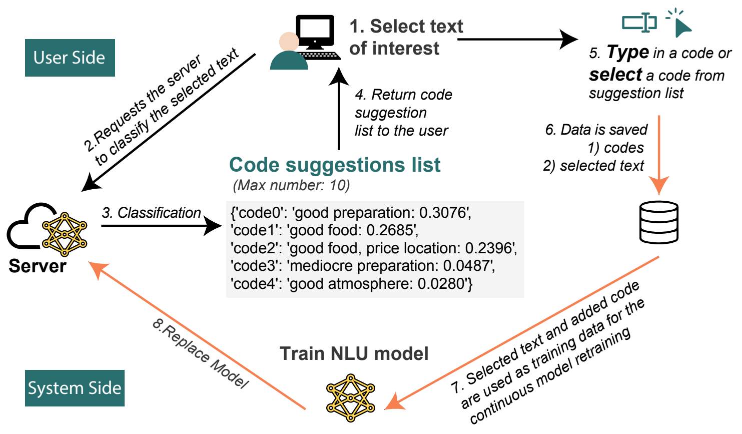
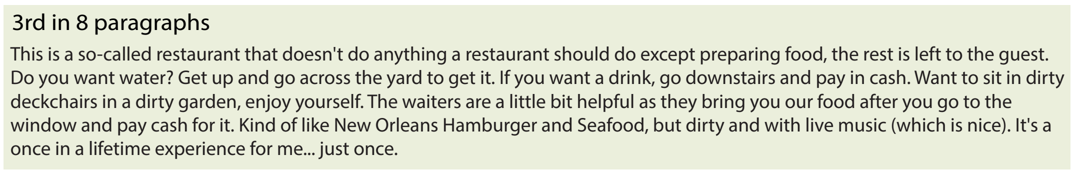

User Side
1. Select text of interest
5. Type in a code or select a code from suggestion list
2.Requests the server to classify the selected text
4. Return code suggestion list to the user
6. Data is saved 1) codes 2) selected text
Code suggestions list
(Max number: 10)
3. Classification
Server
{'code0': 'good preparation: 0.3076', 'code1': 'good food: 0.2685', 'code2': 'good food, price location: 0.2396', 'code3': 'mediocre preparation: 0.0487', 'code4': 'good atmosphere: 0.0280'}
8.Replace Model
Train NLU model
7. Selected text and added code are used as training data for the continuous model retraining
System Side

FIGURE 5.2: Process of recommendation generation. User side: 1) the user selects the text and clicks on "comment" button; the system 2) automatically requests suggestions from the model server, 3) conducts a classification process, 4) returns a list containing up to 10 code suggestions for the user to either select from or refer to, and 5) the user decides to either create their own codes or select one from the list. System Side: 6) the codes and labeled text are subsequently stored for future use, 7) the selected text and added codes are reused as training data to fine-tune a new model, and 8) the updated model is subsequently deployed onto the server.


3rd in 8 paragraphs

This is a so-called restaurant that doesn't do anything a restaurant should do except preparing food, the rest is left to the guest. Do you want water? Get up and go across the yard to get it. If you want a drink, go downstairs and pay in cash. Want to sit in dirty deckchairs in a dirty garden, enjoy yourself. The waiters are a little bit helpful as they bring you our food after you go to the window and pay cash for it. Kind of like New Orleans Hamburger and Seafood, but dirty and with live music (which is nice). It's a once in a lifetime experience for me... just once.

FIGURE 5.3: A sample paragraph for the open coding tasks, extracted and preprocessed from the Yelp reviews dataset.

task comprising eight paragraphs (with an average of 86.8 words per paragraph) was the most suitable length for our study.

Our pilot study also uncovered several typographical and grammatical errors, along with colloquial references that could potentially hinder participants' understanding. To remedy this, we thoroughly cleaned the text before using it for our open coding task. Figure 5.3 depicts one of the revised reviews used in the formal coding tasks.

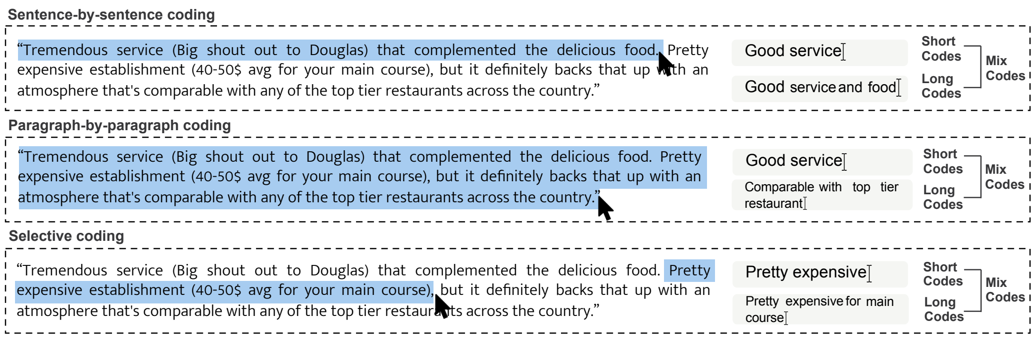
5.4.2 Independent Variables and Conditions

We implemented a split-plot design (Lazar, Feng, and Hochheiser, 2017b) to investigate the effects of two facets of qualitative coding granularity: *Text Granularity* (i.e., unit of analysis or length of text selection) and *Code Granularity* (i.e., length of code in words). The first variable comprises three levels: *Sentence*, *Paragraph*, and *Selective*. The second variable also includes three levels: *Short Codes* (1-3 words), *Long Codes* (4-6 words), and

Mixed Codes (1-6 words). Consequently, this study encompasses a total of $3 \times 3 = 9$ conditions (see Table 5.1 and Figure 5.4).

TABLE 5.1: Nine conditions corresponding to *Text Granularity* (i.e., unit of analysis or length of text selection) and *Code Granularity* (i.e., length of code in words).

		Text Granularity (text length)		
		Sentence (S)	*Paragraph* (P)	*Selective* (E)
Code Granularity (code length)	*Short Codes* (1-3 words) (S)	SS	SP	SE
	Long Codes (4-6 words) (L)	LS	LP	LE
	Mixed Codes (1-6 words) (M)	MS	MP	ME



“Tremendous service (Big shout out to Douglas) that complemented the delicious food. Pretty expensive establishment (40-50$ avg for your main course), but it definitely backs that up with an atmosphere that's comparable with any of the top tier restaurants across the country.”

FIGURE 5.4: Nine Coding Methods.

We opted for a mixed-design approach (i.e., a split-plot design) to ensure the experiment duration remained manageable (approximately 1 hour). We designated *Code Granularity* as a between-subject variable to avoid affecting the participants' coding process, particularly their decision-making regarding labels.

Meanwhile, *Text Granularity* was counterbalanced according to appearance order across various levels, utilizing a Latin Square method (Lazar, Feng, and Hochheiser, 2017b). For each of the three levels of *Text Granularity*, participants were asked to peruse eight selected texts and carry out an open coding task. The impact of differing text selection lengths can be evaluated by comparing conditions within the same row of Table 5.1. Conversely, the effect of varying code lengths can be ascertained by comparing conditions within the same column.

5.4.3 Participants

We conducted our study with 30 participants (12 males and 18 females, mean age = 21.9 years old). We based our participant selection on the following criteria: 1) age of 18 or above, 2) proficient English reading and writing skills, and 3) enrollment in or completion of an undergraduate programme. All participants, being novices in qualitative coding, received appropriate coding training from us prior to the formal study. Each participant received compensation for their time equivalent to 7.25 USD per hour, which aligns with the standard rate approved by our institution's IRB. Additionally, for the follow-up study conducted in section 5.4.9, we recruited 6 participants (3 females, mean age = 26.7 years old) with the same requirements.

5.4.4 Procedure

We partitioned the 30 participants into three groups of 10, each group assigned to propose either short, long, or mixed codes. Independently of their group, all participants underwent the *Sentence*, *Paragraph*, and *Selective* conditions. They were instructed to code each sentence, with the option to skip any that were devoid of meaning (*Sentence*); to assign one code to each paragraph (*Paragraph*); and to code text selections of any length within a paragraph, ranging from phrases to individual or multiple sentences (*Selective*).

Upon signing the consent form, participants were initially briefed on *Open Coding*. This was followed by a 15 to 20-minute training session, introducing them to qualitative coding. Subsequently, they began coding tasks under their assigned conditions. They were also given specific research questions to guide their coding process. Primarily, these queries necessitate participants to discern the customers' opinions and attitudes regarding the store or restaurant presented in the coding material.

To gain insights into users' attitudes towards AI, we implemented a think-aloud protocol during the study. This approach facilitated the observation of participants' coding processes and ensured the tasks were performed correctly. After each study, participants completed a survey and were encouraged to share their reasoning behind the given choices in the survey with the facilitator, and to compare the conditions they experienced. Additionally, we conducted a semi-structured interview at the end of the study to encourage participants to reflect on their experiences.

5.4.5 Dependent Variables

The study aims to investigate the extent to which users trust, rely on, and find the AI system helpful.

Model Performance

To assess the influence of the coding strategies on the model's performance, we employed evaluation metrics from recommendation systems (Tamm, Damdinov, and Vasilev, 2021; *Metrics* n.d.): we consider Precision@k and Mean Average Precision (MAP@k), where k signifies the number of suggestions. To simplify the evaluation, we limit ourselves to the top five suggestions, denoted as $k = 5$.

To establish a "ground truth" for metric evaluation, we utilized the majority of codes stemming from participants' coding outcomes. These predominant codes were then evaluated and ultimately finalized iteratively by two authors, contrasting them with the original text selections. Subsequently, we assigned labels to the AI recommendations generated during participants' open coding process as "1" for relevant and "0" for irrelevant. If the recommendation's meaning from the system corresponds with the "ground truth", it is deemed relevant; otherwise, it is marked as irrelevant. Considering that the original AI-generated suggestions close to 10,000, manually labeling each one is nearly impractical. Hence, two authors independently reviewed 100 AI suggestions using strict criteria for data labeling. Only code suggestions that were directly relevant to the target code or ground truth were deemed relevant. Any discrepancies or ambiguous codes identified were discussed between the two authors. Once they achieved a Cohen's kappa score greater than 0.6, indicating moderate agreement, one of the authors proceeded to label approximately 30% (≈3,000) of the AI suggestions outcomes for each participant's codes.

Specifically, $Precision@k(u) = \frac{|rel(u) \cap rec_k(u)|}{k}$ is calculated as the proportion of relevant recommendations among the top k recommendations provided by the system; $Recall@k(u) = \frac{|rel(u) \cap rec_k(u)|}{|rel(u)|}$ is defined as the proportion of relevant recommendations within the top-k recommendations out of the total number of relevant recommendations. Likewise, mean $AP@k(u) = \frac{1}{|rec_k(u)|} \sum_{i \in rec_k(u)} \mathbb{I}(i \in rel(u)) Precision@rank(u, i)$ is calculated as the average of the Average Precision across all users and requests. This metric incorporates the order information, considering the relevance of items (indicated by $\mathbb{I}(i \in rel(u))$) at their respective ranks (denoted by $rank(u, i)$)[2].

Decision Time

The decision time refers to the duration that users spend on making a decision for each selection, starting from the moment they begin selecting until they finish entering the code. This metric can also serve as an indirect indicator of the difficulty of a coding task (Wright and Ayton, 1988).

Coding Behavior

To enable a thorough comparison across our nine conditions, we examined users' coding behaviors from multiple perspectives, including the number of selections made, the

[2] u is a user identificator; i is an item identificator; $rec_k(u)$ is a recommendation list for user containing top-k recommended items; $rel(u)$ is a list of relevant items for user u from the test set; $rank(u, i)$ is a position of item i in recommendation list $rec_k(u)$; $I[]$ is an indicator function.

length of selections (in words), the length of codes (in words), and the number of unique codes created. The *Coding Behavior* provides insights into coding strategies employed by participants.

Selecting Rate

User reliance (Vorm, 2018; Papenmeier, Kern, Englebienne, et al., 2022) reflects users' willingness to accept system suggestions. We measure users' reliance (Dzindolet et al., 2003; Ashktorab, Desmond, et al., 2021) by $\textit{Selecting Rate} = \frac{\text{Total number of codes selected by users}}{\text{Total number of codes made}}$. The *Selecting Rate* represents the probability of users selecting a suggested code.

Perceived Trustworthiness *and* Perceived Helpfulness

We assess users' *Perceived Trustworthiness* and *Perceived Helpfulness* towards the code suggestions using a five-point Likert scale: 1 = Do not trust at all, 2 = Do not trust, 3 = Neutral, 4 = Relatively trust, 5 = High level of trust. We adapted our questions from prior research that examined users' trust levels towards classifiers and prediction results (Papenmeier, Kern, Hienert, et al., 2022; Rechkemmer and Yin, 2022; Yin, Wortman Vaughan, and Wallach, 2019), including:

1. How much do you trust the *Confidence Score* and *Rank* of the suggestions?
2. How much do you trust the system's ability to include your expected code (*Containing Ability*)?
3. How helpful do you think the suggestions were?

To promote a better understanding of these questions, we verbally illustrate the concepts of the confidence score and ranks to participants as they complete the survey. Furthermore, we provided explicit explanations for each question's intent to participants. For example, we clarified to participants that we were asking whether they believed the confidence score accurately reflected the suggestions' quality; whether they viewed the suggestions' ranking as reliable; and whether they thought the system was capable of producing their expected codes.

Subjective Preferences

We obtained explicit consent from all participants to audio record the entire study. The recorded audio was subsequently transcribed into text format to facilitate further analysis.

5.4.6 Data Analysis

Quantitative analysis

We performed statistical analysis (Norman, 2010) using a mixed two-way ANOVA[3]: we used repeated measures on *Text Granularity*, taking into account the random effect of user. To account for the repeated measures design, we applied appropriate sphericity corrections (Greenhouse-Geisser) when needed, which adjusted both the reported p-values and degrees of freedom when necessary. Post-hoc comparisons were conducted using pairwise t-tests with Bonferroni correction to account for multiple comparisons.

Qualitative analysis

We employed thematic analysis (Braun and Clarke, 2006) to derive themes and groupings for qualitative data. Upon becoming acquainted with the data and establishing initial codes, we organized the transcripts into cohesive themes that aligned with the content. We reviewed the transcripts and audio recordings to extract pertinent quotes corresponding to each identified theme.

5.4.7 RQ1: Impact on Model Performance

The performance of the model is reported in Table 5.2.

The results suggest that *Code Granularity* does not exert a significant influence on Precision@5, MAP@5 metrics, but a main effect on Recall@5 ($F_{2,27} = 3.79, p = .035$). Specifically, *Mixed Codes* ($M = 0.31$) exhibits a lower Recall@5 compared to *Short Codes* ($M = 0.48, p = .022$).

Text Granularity significantly impact Precision@5, Recall@5, and MAP@5 metrics ($p < .05$ for all). For MAP@5, we noted a trend where models performed consistently worse under *Sentence* ($M = 0.36$) than *Selective* ($M = 0.46, p < .01$) and *Paragraph* ($M = 0.47, p < .01$). Similarly, for Recall@5, models performed consistently worse under *Sentence* ($M = 0.38$) than *Selective* ($M = 0.47, p < .025$). Our findings also suggest no interactions between *Code Granularity* and *Text Granularity*.

Summary

The lower Recall@k for *Mixed Codes* compared to *Short Codes* suggested that less relevant code suggestions are likely to be extracted from possible candidate sets because of the uncontrolled length of codes. Moreover, the model, when tasked on *Paragraph*, demonstrably surpasses *Sentence*. The performance boost may be attributed to the verbose text in the *Paragraph* configuration, which provides a richer dataset for the

[3]https://pingouin-stats.org/generated/pingouin.mixed_anova.html

TABLE 5.2: The model's performance. All values fall within the range of 0 to 1.

Factor1: Code Granularity	Factor2: Text Granularity	Precision@k (M ± S.D.)	Recall@k (M ± S.D.)	MAP@k (M ± S.D.)
Short Codes (1-3 words)	Sentence	0.19 ± 0.07	0.53 ± 0.19	0.38 ± 0.11
	Paragraph	0.21 ± 0.10	0.38 ± 0.18	0.46 ± 0.24
	Selective	0.20 ± 0.06	0.53 ± 0.20	0.46 ± 0.08
Long Codes (4-6 words)	Sentence	0.21 ± 0.04	0.34 ± 0.20	0.38 ± 0.12
	Paragraph	0.26 ± 0.11	0.47 ± 0.19	0.53 ± 0.17
	Selective	0.28 ± 0.09	0.61 ± 0.30	0.50 ± 0.19
Mixed Codes (1-6 words)	Sentence	0.17 ± 0.04	0.28 ± 0.07	0.32 ± 0.06
	Paragraph	0.22 ± 0.08	0.38 ± 0.17	0.43 ± 0.16
	Selective	0.21 ± 0.07	0.28 ± 0.07	0.41 ± 0.13

classifier. However, shorter text selections might inherently convey less information, thereby possibly reducing the probability of a match between users' codes and the selected text. In contrast, longer selections could offer more context, potentially leading to more accurate model classifications. However, this remains speculative, and we seek further validation through users' subjective feedback.

5.4.8 RQ2: Impact on Decision Time and Coding Behavior

Coding Behavior

The *Coding Behavior* results are outlined in Table 5.3. Not all comparisons bear logical consistency—for instance, comparing the length of selections in *Paragraph* and *Sentence* conditions. Further intriguing observations surface when we examine (1) the number and length of selections for *Mixed Codes*, *Short Codes*, and *Long Codes* under the *Selective* condition and (2) the difference in code length between *Mixed Codes* vs. *Short Codes* and *Mixed Codes* vs. *Long Codes*. Specifically:

(1) Under *Selective* condition, the length of selections in *Long Codes* ($M = 29.22$) significantly surpasses that in *Mixed Codes* ($M = 12.07, p < .001$) while no difference between *Mixed Codes* ($M = 12.17$) and *Short Codes* ($M = 12.07$). The number of selections shows no significant difference.

(2) In the length of code, *Long Codes* ($M = 5.05$) is longer than *Mixed Codes* ($M = 3.37, p < .001$), and *Mixed Codes* ($M = 3.37$) is longer than *Short Codes* ($M = 2.19, p < .001$).

(3) In terms of code length, there is no significant difference observed between the codes in *Selective* ($M = 3.13$) and codes *Sentence* ($M = 3.19$).

Decision Time

Decision Times across all conditions are shown in Figure 5.5.

TABLE 5.3: Summary of Coding Behavior. Among nine conditions, *Mixed Codes* × *Selective* is the baseline, having little to no code constraints and thus closely representing open coding.

Factor1: Code Granularity	**Factor2: Text Granularity**	**Coding Behavior**		
		Number of Selection[a] (M±S.D.)	**Length of Selection[b] (M±S.D.)**	**Length of Code (M±S.D.)**
Short Codes (1-3 words)	Sentence	–	–	2.06 ± 0.54
	Paragraph	–	–	2.51 ± 0.59
	Selective	29.70 ± 14.09	12.17 ± 12.37	2.00 ± 0.54
Long Codes (4-6 words)	Sentence	–	–	4.88 ± 0.92
	Paragraph	–	–	5.34 ± 0.72
	Selective	15.90 ± 7.37	29.22 ± 20.71	4.93 ± 0.93
Mixed Codes (1-6 words)	Sentence	–	–	2.63 ± 1.26
	Paragraph	–	–	5.03 ± 1.08
	Selective (baseline)	**28.30 ± 12.51**	**12.07 ± 9.71**	**2.46 ± 1.26**

[a] The number of selections for *Paragraph* is consistently 8, while for *Sentence* is consistently around 35.
[b] The selection length for *Paragraph* consistently averages around 87 words, while for *Sentence*, it typically averages around 14.6 words.

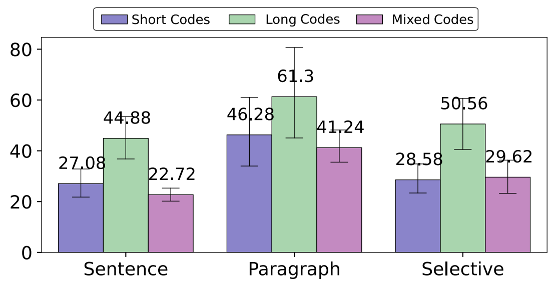


FIGURE 5.5: Average Decision Time (Seconds). The time needed to make a decision for each selection. Final results for *Selecting Rate* and *Decision Time*. Error bars represent .95 confidence intervals.

Code Granularity. A significant main effect of *Code Granularity* on *Decision Time* was detected ($F_{(2,24)} = 11.13, p < .001$). Participants tended to spend more time formulating *Long Codes* ($M = 52.2s$) compared to *Short Codes* ($M = 34.0s, p = .014$) and *Mixed Codes* ($M = 31.2s, p < .01$).

Text Granularity. A significant main effect of *Text Granularity* on *Decision Time* was observed ($F_{(2,48)} = 10.13, p < .001$). Generally, participants required more time to label *Paragraph* ($M = 52.2s$) in comparison to *Selective* ($M = 35.6s, p = .023$) and *Sentence* ($M = 31.6s, p < .001$).

Interactions. No significant interaction was detected ($p = .97$).

Summary

Decision Time and Task Difficulty Participants found creating *Long Codes* more challenging due to a four-word minimum, unlike the one-word minimum for *Short Codes* and *Mixed Codes*. Similarly, *Decision Time* increased with *Text Granularity*, with longer coding periods for *Paragraph*, implying greater task difficulty and time commitment for individual coding selection tasks.

Sentence $\approx$ Selective; Mixed Codes $\approx$ Short Codes Despite certain disparities, participants demonstrated similar coding behavior across both the *Selective* and *Sentence* conditions for code length, as well as between *Mixed Codes* and *Short Codes* for number and length of selections.

Correlation between Length of Codes and Text Crafting longer code names often necessitated larger text selections, a fact further corroborated by the text length between *Long Codes* and *Short Codes* under *Selective* condition, while the number of selections significantly decreased in longer codes conditions, while the number of selections in *Short Codes* was twice as many.

5.4.9 RQ3: Impact on User Reliance

In this section we examined the users' reliance on AI, we are specifically concerned about the 1) relationship between AI model performance and users' reliance. 2) whether there is a risk of overreliance that could potentially impact coding quality.

Selecting Rate

The *Selecting Rate* are visualized in Figure 5.6. Our statistical evaluation reveals that *Text Granularity* has a significant influence on *Selecting Rate* ($F_{(2,54)} = 15.838, p < .001$), whereas *Code Granularity* does not demonstrate a main effect. Pairwise differences are detected (all $p < .05$): *Selective* registered the highest *Selecting Rate* (with a mean of $M = 32\%$ across conditions), followed by *Sentence* (with a mean of $M = 26\%$), and lastly *Paragraph* (with a mean of $M = 16\%$). Moreover, a notable interaction between *Text Granularity* and *Code Granularity* was detected ($F_{(4,54)} = 2.766, p = .036$).

Solely selecting: when users only select codes from AI suggestions. The statistical analysis indicates that *Code Granularity* significantly influences "solely" selecting rate ($F_{2,27} = 3.887, p = .032$). Similarly, *Text Granularity* has a main effect on the "solely" selecting rate ($F_{2,54} = 22.117, p < .001$). Moreover, both *Code Granularity* and *Text Granularity* exhibit an interaction effect ($F_{4,54} = 3.472, p = .014$). We observed distinct pairwise differences, all at $p < .001$: Users exhibited selection behavior more frequently with *Sentence* and *Selective* than with *Paragraph*; there was no distinguishable difference between *Sentence* and *Selective*. When employing *Short Codes* for coding, these tendencies became even more pronounced.

Selecting and modifying: when users select and modify codes from AI suggestions. The statistical evaluation demonstrates that *Code Granularity* has a notable impact on the "selecting and modifying rate" ($F_{2,27} = 3.375, p = .049$), while *Text Granularity* does not. When comparing pairs, it's evident that users tend to first select and subsequently modify the code more frequently under the *Long Codes* condition compared to the *Short Codes* condition ($p = .046$).

Therefore, user reliance on AI is more pronounced in the *Selective* condition, where users are tasked with selecting only pertinent text portions—a situation that closely resembles real-world coding scenarios. On the contrary, the *Paragraph* condition had the lowest *Selecting Rate*, particularly with *Short Codes*, likely due to the difficulty of summarizing and coding an entire paragraph with a 3-word limit. This demanding task caused both AI and participants to struggle. On the other hand, the length of the codes influences users' coding and modification habits. Specifically, longer codes often serve as an "indirect" reference, prompting users to first select and then adjust the codes, which might have enhanced user's subjective feelings on these conditions as reported in RQ 5.4.10 and RQ 5.4.11.

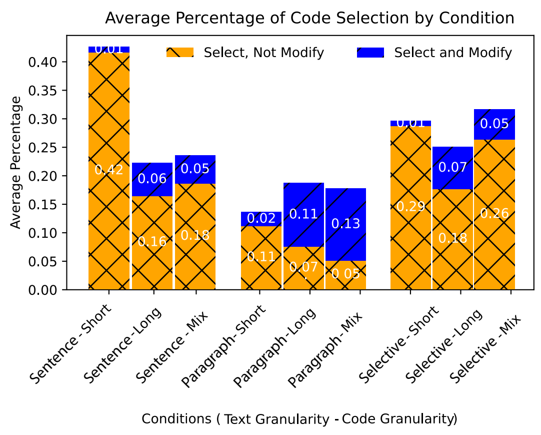


FIGURE 5.6: *Selecting Rate* (0-1). Users' receptiveness to code suggestions produced by the system. Final results for *Selecting Rate* and *Decision Time*. Error bars represent .95 confidence intervals.

Over-reliance concerns

Given some high *Selecting Rate* in conditions like *Short Codes* × *Sentence*, *Mixed Codes* × *Selective*, and *Long Codes* × *Paragraph*, we are concerned that some coders may have overly relied on the system under these conditions, thereby affecting the final coding quality. This concern is elevated when observing the specific *Selecting Rate* for each coder: 6 out of 30 participants demonstrated a *Selecting Rate* above 50% in *Selective* conditions. When reviewing the *Selecting Rate* of these participants under the *Sentence* conditions, a similar trend emerged.

Comparing the Coding Results With and Without AI Assistance

Supplementary Study In order to validate our concern, we carried out a supplementary study. This involved 6 more participants performing coding tasks without AI assistance under conditions suspected to foster over-reliance. The procedure is identical to that of our primary study.

The experimental setup encompasses three previously mentioned conditions that demonstrated the highest *Selecting Rate*. Due to the analogous behavior observed between users for *Selective* and *Sentence*, *Mixed Codes* and *Short Codes* (see section 5.4.8), we anticipate that an analysis of the *Short Codes* × *Sentence* and *Mixed Codes* × *Selective* condition would probably yield results comparable to those from the *Short Codes* × *Selective* condition.

Results We decided to perform a qualitative comparison between the final quality of the coding results with and without AI assistance. The results are detailed in Table 5.4.

We observed that codes in *Short Codes* $\times$ *Sentence* with AI assistance seem to have fewer code variances than coding without AI, even though their primary category is similar. For instance, 'bad food' is a code with AI assistance, while codes such as 'cold, old fries' or 'dislike burger set' serve as analogs of 'bad food' without AI assistance, albeit with more variance. When assisted by AI, users might be presented with a 'bad food' suggestion, which they may subsequently adopt instead of proposing other expressions.

The *Short Codes* $\times$ *Sentence* condition appears 'acceptable', with participants exhibiting a relatively commendable *Selecting Rate*. Likewise, the *Mixed Codes* $\times$ *Selective* condition reveals a similar change in code results. We might anticipate similar decreases in other conditions like *Short Codes* $\times$ *Selective*. Interestingly, the *Long Codes* $\times$ *Paragraph* condition seems to demonstrate a relative consistency, regardless of the presence or absence of AI.

Indeed, the decrease in code variance, to a certain extent, could be perceived as beneficial since it might result in more focused coding and reduce the effort needed to group variances. **However, it risks yielding coding outcomes that appear less substantial and somewhat superficial. This could potentially influence the discussion and creation of a codebook in subsequent stages of real qualitative analysis.**

5.4.10 RQ4: Impact on Perceived Trustworthiness and Helpfulness

Perceived Trustworthiness

Results of users' self-reported trustworthiness are depicted in Table 5.5 and Figure 5.7. Although we anticipated variations in users' perceptions of trustworthiness across different conditions, we observed no significant main effects on *Perceived Trustworthiness* in any of the measures. Overall, users held neutral attitudes ($score \approx 3$) towards the system's components and its ability to suggest their desired codes. However, the most intriguing findings predominantly stem from the *Perceived Helpfulness*.

Perceived Helpfulness

Code Granularity A significant main effect of *Code Granularity* on *Perceived Helpfulness* were observed ($F_{(2,27)} = 9.789, p < .001$). The system's suggestions were deemed more helpful by users in the *Long Codes* condition ($M = 3.97$) than those in the *Mixed Codes* condition ($M = 2.53, p < .001$). Likewise, in the *Short Codes* condition, the system was perceived as more helpful ($M = 3.20$) than in the *Mixed Codes* condition ($M = 2.53, p = .049$).

Text Granularity No significant main effect on *Perceived Helpfulness* was discerned.

TABLE 5.4: Comparison of typical coding results for each condition (LP = *Long Codes* × *Paragraph*, SS = *Short Codes* × *Sentence*, and ME = *Mixed Codes* × *Selective*), both with and without the application of AI. Each cell presents codes derived from a typical user's results under the specified condition. While the "with AI" condition may yield codes with a higher selection rate, they may lack nuanced detail. In contrast, the "without AI" condition tends to generate more detailed codes.

	With AI	Without AI
LP	**Food Quality:** cheap good dessert bad breakfast decoration overall bad food try another venue good Thai food very happy meal lazy service decent food won't return **Service and Cleanliness:** self-service dirty restaurant won't visit again good food nice people recommended visit lazy service decent food won't return	**Food Quality:** cheap good location dessert bad breakfast good server ok pizza bad burger nice people good food have games tasty coconut soup and pad thai good promotion friendly people tasty food lazy service good food average pricing **Service and Cleanliness:** poor service dirty live music avail clean good service tasty reasonable price lazy service good food average pricing
SS	**Food Quality:** cheap food bad food ok food good food expensive food quality dropped **Service:** good service bad service ok service **Ambiance/Environment:** bad decoration good music good entertainment dirty **Others:** good offer	**Food Quality:** feels cheap good food, service neutral food bad food dislike burger set cold, old fries pricey pizza okay pizza good food people coconut soup pad thai same menu tried liked coconut soup coconut soup creamy good pad thai good peanuts, noodles good chicken good hot wings lots of sauce delicious sushi, affordable affordable sushi freshness and variety **Ambiance/Atmosphere:** jaded decor new orleans vibe quiet, competent chef **Service:** poor service recovery (2) decent server lack of service water not served decent service friendly staff order mixup Recommendations and Reviews: positive recommendation (2) mixed review **Customer Relationship:** better previous experience lost customer (mentioned twice) purchase inconvenient recent customer overall satisfied potentially lost customer **Others:** not clean beer with brother played nintendo prefer fewer herbs near hotel, convenient promo good returning for pizza large party, hibachi
ME	**Food Quality:** food tastes bad food tastes normal tasty fresh food with huge variety good place food and people **Service:** poor service good service cheap but poor food and service **Pricing:** expensive worth worth and tasty expensive and tasty tastes and feels cheap **Others:** will not try this again unhygienic quiet	**Food Quality:** cheap good price location and dessert bad burger bad fries pricey but ok pizza good coconut soup good pat thai good hot wings great sushi and reasonable price fresh with huge variety bad service good food but expensive **Service:** bad services poor service (mentioned twice) good server nice and friendly clean environment and good service great place, food and people **Attitude and others** will not come again dirty environment good location recommended 1 for 1 offer bad service good food but expensive

TABLE 5.5: Summary of Values of *Perceived Trustworthiness* and *Perceived Helpfulness*. All DVs are on a Likert scale from 1 to 5.

Factor1: Code Granularity	**Factor2: Text Granularity**	**Perceived Trustworthiness**		**Perceived Helpfulness (M±S.D.)**
		Confidence Score, Rank (M±S.D.)	**Containing Ability (M±S.D.)**	
Short Phrases (1-3 words)	Sentence	2.95 ± 1.15	3.00 ± 1.25	3.60 ± 1.07
	Paragraph	2.75 ± 1.13	2.20 ± 1.34	2.30 ± 0.95
	Selective	3.45 ± 0.90	3.30 ± 1.16	3.70 ± 1.16
Long Phrases (4-6 words)	Sentence	3.20 ± 0.97	3.60 ± 0.97	3.80 ± 1.32
	Paragraph	3.50 ± 0.84	3.40 ± 1.17	4.30 ± 1.16
	Selective	3.50 ± 1.07	3.60 ± 0.84	3.80 ± 1.31
Mix Phrases (1-6 words)	Sentence	2.35 ± 1.15	2.40 ± 1.35	2.70 ± 1.16
	Paragraph	3.05 ± 0.97	3.40 ± 0.97	3.50 ± 0.71
	Selective	**2.80 ± 0.98**	**3.20 ± 1.14**	**1.40 ± 0.97**

Interaction effects A noteworthy interaction was detected between two factors on *Perceived Helpfulness* of the system ($F_{(4,54)} = 7.94, p < .001$). Particularly, under the *Mixed Codes* condition, the system was rated significantly more helpful in the *Paragraph* condition compared to the *Selective* condition ($p < .001$).

For descriptive statistics, the mean *Perceived Helpfulness* scores exceeds 3 (refer to Figure 5.7), albeit experiencing slight reductions under particular conditions such as *Mixed Codes* × *Selective*, *Mixed Codes* × *Sentence*, and *Short Codes* × *Paragraph*. We also observed that pairing *Long Codes* with a high level of *Text Granularity* (*Paragraph*) resulted in the highest mean *Perceived Helpfulness* (4.3/5), significantly surpassing the scores in any of the other eight conditions. Unexpectedly, the baseline condition (*Mixed Codes* × *Selective*) results in the lowest *Perceived Helpfulness*. Moreover, when *Short Codes* is paired with *Paragraph* coding, it results in significantly diminished *Perceived Helpfulness*, with users rating the system as not helpful. We delve deeper into this phenomenon from the perspective of task difficulty in Section 5.5.

5.4.11 RQ5: Impact on Subjective Preferences

In this section, we encapsulate the feedback conveyed by participants during and subsequent to the study.

User Preferred Selective

Greater control over the selection enabled participants to receive more accurate suggestions, which subsequently motivated them to choose suggestions more frequently. This is evidenced by the participant comment: *"Because I can adjust the selection, then I think the way the numbers (confidence score) work...sometimes the one on the top is the one I want."* (P26, *Mixed Codes* and *Selective*).

Even though the length of text selections was similar between *Selective* and *Sentence*, participants showed a preference for *Selective*, as articulated by P18: *"The main difference is that for the sentence one, some sentences don't have meaning. But for selective, I could group the sentences with the same meaning together under one topic."* (*Long Codes* and *Selective*).

Imperfect AI Suggestions Still Contribute Value

In many instances, participants found the suggested codes were close to their original ideas: *"I find it relatively helpful. The recommended codes bear some similarity to what I had in mind, so I don't have to ponder excessively."* (P20, *Long Codes* and *Paragraph*).

Whereas, even if the suggestions didn't always precisely align with the participants' requirements, they were still viewed as helpful. The suggested codes from the list inspired participants to combine existing codes to generate new ones and refine their

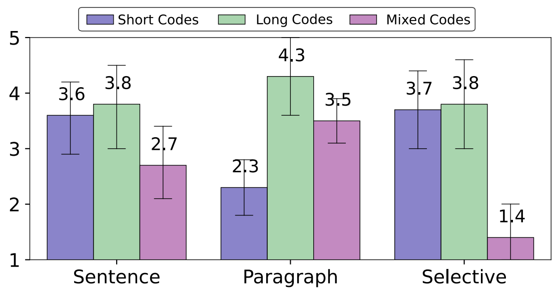


FIGURE 5.7: User's *Perceived Helpfulness* of code suggestions. Error bars show .95 confidence intervals. Y-axis represents 1-5 Likert score, where 1 represents a complete lack of helpfulness and 5 is the highest level of helpfulness.

own. The system's feature that allowed users to modify suggested codes was particularly appreciated:

"I think they [suggestions] are quite helpful. They kind of give you a hint about what you could write for the keywords or summaries." (P28, *Mixed Codes* and *Sentence*).

Code Suggestions Promote Consistency

Several participants highlighted that the suggestions aided them in maintaining consistency throughout the coding process:

"The recommendation list seems somewhat helpful because it enables me to apply a consistent metric when assessing these text streams. As a result, I can establish a bit more consistency between the texts as I formulate my codes." (P22, *Mixed Codes* and *Sentence*).

Too Long Text Selections (Paragraph) Presents Challenges

Overall, participants indicated that AI would need to bolster its performance to meet their expectations. Specifically, a notable challenge inherent in AI is its struggle to capture nuanced information within the text:

"The system failed to capture the context of sentences within the paragraph. At times, sentences were unrelated to one another - one might discuss good service while another addressed food, indicating different contexts within each review. Consequently, the system couldn't discern the nuances of individual sentences and provide accurate confidence scores." (P39, *Mixed Codes* and *Sentence*).

5.5 Discussion

Our discussion first delves into an examination of task difficulty, outlining how various conditions were deemed more challenging than others. Following this, we unpack the diverse elements of trust and reliance between humans and *AIQCs* under evaluation. Lastly, we traverse through the significance of differing granularity conditions, especially in relation to their relevance in more realistic qualitative analysis scenarios.

5.5.1 Task Difficulty Across Conditions for *Open Coding*

Qualitative Open Coding: A Series of Distinct Tasks Rather than a Singular Whole

In essence, our nine conditions simulate various levels of difficulty associated with *Open Coding* tasks. Our findings advance a nuanced understanding, positing that coding tasks can differ based on their intrinsic difficulty or the effort demanded. Some tasks may boost the model's performance, while others might hinder it. This viewpoint diverges from prior research in this domain, which might have predominantly treated *Open Coding* as a uniform, undifferentiated task, intending to devise a solitary method to facilitate it. This view has overlooked the inherent complexity of subjective tasks like qualitative coding, a scenario where human-AI interaction could play a pivotal role.

Challenging Paragraph Conditions

In *Paragraph*, the units of text to be coded were longer compared to those in the *Sentence* and *Selective* conditions. Moreover, based on subjective feedback, *Paragraph* might have included various contradictory nuanced information. Therefore, participants needed more time to decide on a code, leading us to infer that the coding task under the *Paragraph* conditions was relatively more challenging.

However, according to the model performance data, an increase in context and text selection seems to enable the model to get higher precision. Conversely, a decrease in the text selection did not yield the same level of model performance.

At first glance, they may seem contradictory to each other. However, a paragraph may have a higher probability of matching the code, but it may also contain extraneous and even contradictory information not included in the “highly matching code". Consequently, despite the possibility of paragraphs yielding higher model performance scores, they present a substantial challenge to both users and AI, particularly due to the nuanced information they may encapsulate.

The Complexity of Long Codes Compared to Short Codes and Mixed Codes

The *Long Codes* conditions seemingly posed greater challenges for participants due to the requirement of a minimum number of words for each code, as indicated by the extended decision-making times. Conversely, participants found the *Short Codes* conditions less strenuous as they closely mirrored conventional coding scenarios, with code lengths similar to those in the *Mixed Codes*.

5.5.2 Reliance and Perceptions Discrepancies Due to Varied Task Difficulties

Higher Reliance for Simpler Tasks

In terms of *Text Granularity*, our data reveal that users exhibit greater reliance on AI, in simpler tasks at the *Selective* and *Sentence* levels, as indicated by an average selection rate of 29%. This contrasts with the harder *Paragraph* tasks, which only saw an average selection rate of 17%.

Significant individual differences also emerged. For instance, in *Selective* and *Sentence* conditions, certain participants (P14 for *Selective*, P7 for *Sentence*) ignored all suggestions, while others had high selection rates (73% for P10 in *Selective*, 53% for P17 in *Sentence*). For the tougher *Paragraph* conditions, an overwhelming 11 participants completely disregarded the system's suggestions, with the peak selection rate merely reaching 38% (P25, P27, P28).

Contrasting Reliance and Perceived Helpfulness in Complex Tasks

Interestingly, while reliance was at its nadir in the *Paragraph* setting, *Perceived Helpfulness* was substantially high, and *Perceived Trustworthiness* was also considerable, especially for more complex tasks with longer AI suggestions. In particular, during tasks with longer codes (*Long Codes* $\times$ *Paragraph* and *Mixed Codes* $\times$ *Paragraph*), users, while finding individual suggestions inadequate for selection (thus the low *Selecting Rate*), still referenced them or made minor adjustments to construct their codes. This added flexibility enhanced their *Perceived Helpfulness*, especially in the challenging task of *Long Codes* $\times$ *Paragraph*, scoring 4.3/5 in *Perceived Helpfulness*.

5.5.3 Over- and under-reliance on AIQCs

As discussed, different coding tasks resulted in varying reliance on the system. However, *AIQCs* should strike a balance between user exploration and AI assistance, without promoting either an excessive reliance on or insufficient use of AI suggestions. Over-reliance can lead to shallow coding, while under-reliance may result in missed opportunities for valuable AI assistance. As such, striking the right balance is crucial for the effective use of AI in qualitative coding.

Reasons for Under-reliance

The low *Selecting Rate* for *Paragraph* tasks is primarily due to fewer coding units, resulting in fewer data points for model training. Typically, participants would only choose from the last few suggestions, with a total selection count of approximately 8, in comparison to around 35 in *Sentence* and 20+ in *Selective* tasks. Thus, we anticipate that with more data points to train the model, the system reliance would increase.

For those scenarios where data points may not significantly increase, enabling users to edit their codes post-selection could offer indirect assistance (through the process of selection and then editing). Although these improvements might not be prominently reflected in *Selecting Rate*, they could elevate users' subjective experience, potentially bolster the system's trustworthiness, and encourage users to fully exploit the system (BANOVIC et al., 2023).

Over-reliance Risk

As detailed in section 5.4.9 and 5.4.9, reliance could ostensibly reduce human effort by enabling users to re-utilize previous codes, causing a more focused coding. However, it also carries an over-reliance risk. Over-reliance might disrupt the delicate balance between focused coding and the generation of diverse coding outcomes, which could narrow the scope of interpretation and potentially reduce the depth and breadth of the coding process. Therefore, ensuring a balance between AI assistance and human input is key to maximizing both the efficiency and depth of qualitative coding. This balance should never be underestimated in the design of a truly trustworthy *AIQCs*, as opposed to a system that deceives users' trust without meriting it (BANOVIC et al., 2023).

5.5.4 Optimal Code Granularity Varies Between Users and AI

Participants usually generate shorter codes when possible. The *Mixed Codes*, which allows them to create more specific and longer codes, resembles real-life open coding tasks more closely. The similar code lengths in *Short Codes* and *Mixed Codes* under both *Selective* and *Sentence* conditions imply that participants aim to minimize their codes' length when given the option in the given study. Hence, *Short Codes* or *Mixed Codes* can be considered optimal code granularity for users.

Conversely, while users prefer to add shorter codes, they anticipate longer code suggestions from AI. This is evident in the consistently higher perceived helpfulness of *Long Codes* over other *Text Granularity* conditions. We infer that longer AI suggestions enable more expressiveness, thereby reducing potential misinterpretations between users and AI.

Moreover, a discrepancy exists between human and AI preferences when it comes to text selection for coding. Participants generally favored labeling shorter selections that accommodated shorter codes, as indicated by the similar selection lengths in the *Short Codes* × *Selective* and *Mixed Codes* × *Selective* cases, where users selected only the critical single semantic elements for coding. On the other hand, AI favored a more comprehensive context for accurate code prediction, thus creating a divergence between human and AI inclinations. In particular, the *Short Codes* × *Paragraph* condition exemplified a considerable mismatch. Although it offers the optimal code length for users, the restriction of a three-word code for an extensive text led to a disconnect between the text and code, significantly increasing the task difficulty. This resulted in the lowest *Selecting Rate* of 14% among users and lower *Perceived Helpfulness* than neutral (2.3/5).

In addition, users seemed to favor uniform AI suggestions, as indicated by their perception of *Mixed Codes* suggestions as less helpful than both *Long Codes* and *Short Codes*. The mix of long and short codes in the suggestion list under *Mixed Codes* might have led to information overload, making it challenging for users to locate useful information. Moreover, while the added flexibility, notably in *Selective* codes, could theoretically benefit the users, it seemed to inadvertently decrease the user's perceived helpfulness of the suggestions in *Mixed Codes* × *Selective*.

5.5.5 Coding Strategies in Real Life

Selective is Best for Coding

Notably, we observed the highest levels of *Behavioral Trust* in the *Selective* coding conditions. *Selective* coding most accurately emulates how a single researcher might begin to navigate data in a real-life scenario. They would select the most pertinent phrases and then generate a suitable label. In practice, it is crucial to utilize various coding levels, alternating perspectives, and varying depths of understanding to produce a more comprehensive and diverse range of codes.

Sentence for Collaborative Coding

In fact, *Selective* and *Sentence* coding scenarios share several similarities, notwithstanding certain notable differences. The variance between these two conditions is significantly less than that between them and the *Paragraph* condition. While *Selective* may generally be the go-to granularity for coding, *Sentence* level coding can be particularly beneficial for collaborative coding, where consistency between multiple users is required. This is especially important when computing inter-rater reliability scores, as it requires a straightforward, unambiguous text selection unit.

Paragraph for Summarizing Long Texts

Paragraph may still prove valuable for users attempting to summarize lengthy texts in real coding scenarios. Opting for a *Paragraph* approach assists in distilling entire pages into a few concise labels.

5.6 Implications for Design

We propose several guidelines to cultivate appropriate reliance and foster a productive human-AI collaboration within the context of *AIQCs*.

5.6.1 Fostering Trustworthiness during Under-reliance on *AIQCs*

Offering Extensive and Modifiable Suggestions

We observed that while users generally found less difficulty in creating *Short Codes* and *Mixed Codes*, *Long Codes* suggestions seem to be perceived as more beneficial and trustworthy. The utility of longer suggestions stems from their capacity to convey a wealth of information, thereby minimizing ambiguity and potentially delivering deeper meaning. This extensive nature also enables users to refine their code by editing suggestions, tailoring them to their unique requirements. This active participation makes users feel more in control, which could result in increased trust and system usage.

Exploiting Larger Training Datasets

We noted that some participants did not utilize AI suggestions during the *Paragraph* tasks. This lack of use could be attributed to the reduced quantity and quality of the data used for training, resulting in initially subpar AI suggestions.

To address this concern, we propose the application of data augmentation techniques[4], generating additional training data. Furthermore, if feasible, the integration of data from diverse users and sources could be beneficial for open coding. This recommendation aligns with the current trend towards a data-centric approach, as advocated in recent literature (Eyuboglu et al., 2022; Motamedi, Sakharnykh, and Kaldewey, 2021; Whang et al., 2021).

Facilitating Open Coding Through Multifaceted Models

We further recommend utilizing multiple models to generate code outputs from diverse perspectives. Rather than exclusively relying on text classification or topic modeling, we advocate for considering and integrating other methodologies, such as Generative AI like GPT-4 [5]. By doing so, users can construct their codes under a wider umbrella of system assistance, thereby enabling more informed decision-making (Feuston and Brubaker, 2021; J. A. Jiang et al., 2021). Moreover, the system could offer suggestions inspired by codes from other users, thus presenting an alternate view of the data.

[4]`https://www.tensorflow.org/tutorials/images/data_augmentation`

[5]https://atlasti.com/ai-coding-powered-by-openai, https://openai.com/chatgpt

5.6.2 Mitigating Over-reliance to Prevent Shallow Codes

We've recognized the potential for over-reliance in certain situations. At times, AI that lacks sufficient trustworthiness could deceive users into considering it 'trustworthy' (BANOVIC et al., 2023), leading to excessive reliance. Below, we delve into several specific design strategies to mitigate this issue.

Implementing a Delay in Suggestions Display upon Selection.

The system could be designed to deliberately delay the display of suggestions or only present codes upon a user's request, ensuring they appear specifically when a user struggles to formulate a code (Buçinca, Malaya, and Gajos, 2021). This feature would afford the user sufficient time to contemplate an initial code, and subsequently ensure that the displayed suggestions align effectively with their requirements.

Providing Explanations for AI Suggestions

A promising strategy might be to present explanations alongside the code suggestions (Vasconcelos et al., 2023). For example, by displaying the original data from which the suggestions are derived, coders can compare and ascertain the appropriateness of coding the current data under a specific code. This approach not only encourages deeper thinking but also fosters appropriate reliance on the system.

5.7 Limitations and Future Work

This work has limitations. First, understanding the accuracy and overall performance of the model is crucial for gauging the system's effectiveness under varying conditions. Ideally, each text segment should have a corresponding ground truth value against which we can compare system recommendations to evaluate model performance.

To estimate this, we utilized the majority of user codes as a proxy for the 'ground truth' of a specific text segment. However, this approach offers only an approximation of the actual scenario. It might overlook the nuances of coding; for instance, an insightful interpretation that slightly deviates from the core idea could still be correct. However, in our labeling process, it might be deemed "not relevant", which could undervalue the system's suggestions compared to their true potential usage. Future research could further investigate better ways to evaluate model performance or establish a more suitable ground truth for subjective tasks such as qualitative coding.

Moreover, our choice to focus on codes of varying abstraction levels and specificity stems primarily from their representation of different user coding habits has been stated in Section 5.2.2. In particular, to emulate these different levels of abstraction and interpretation, we opted for a simplistic albeit imperfect method for user operation. Codes of three words or less represent concise coding (short codes), those between four to six words signify verbose coding (long codes), and codes ranging from one to six words (mixed codes) represent natural coding. We selected these parameters for the study setup based on pilot tests conducted on our own materials prior to the formal study. However, we acknowledge that this classification has its shortcomings. For instance, the specific length of the codes is intimately linked to the domain of the coding material, and the delineation of codes across different levels of the factor remains unclear. Looking forward, it would be promising to extend these results to various types of content, with the goal of gaining a more comprehensive understanding of the specific assistance and suggestions users truly need.

Additionally, the selected parameters of limitation (1-3 words, 4-6 words, etc.) used for coding are, while simplistic, imperfect representations of user operations, mirroring the range from concise to lengthy and natural coding habits. The specific values, however, could vary significantly based on the coding material. Moreover, it's essential to motivate participants to execute tasks with greater efficiency, thereby achieving a more precise measure of decision-making time. Future studies should further investigate these aspects, aiming to devise more general strategies for controlling and managing human-AI interaction habits.

Overall, our primary objective in this work is to appeal to developers and researchers, underlining the importance of developing trustworthy *AIQCs* that fosters robust human-AI collaboration by taking into account the unique dynamics of human-AI interaction within qualitative coding. It is critical to not only integrate advanced technologies into this domain but also to view *Open Coding* as a collection of different subtasks. Therefore, the design of various tools should aim to support the nuanced and varied coding tasks inherent within *Open Coding*. Furthermore, the potential risks for under-utilization (under-reliance) and over-reliance should be considered, as the former could result in the system being under-utilized, and the latter might lead to less insightful coding outcomes.

5.8 Conclusion

In this work, we explored how *Code* and *Text Granularity* could influence user trust and reliance in *AIQCs* by conducting a split-plot design study with 30 participants and a follow-up study with 6 participants. Our study highlighted that *Open Coding*, due to its unique human-AI interaction dynamics, should be approached as a composite of various subtasks. Each of these subtasks necessitates a tailored design. Our findings also indicate trust discrepancies stemming from varied subtask difficulties and illuminate the problems of over-reliance and under-reliance existing in different conditions. These results form a foundation for future research on the user trust, reliance, and utility of *AIQCs*.

Chapter 6

Building A Lower-barrier, Rigorous Workflow for Collaborative Qualitative Analysis with Large Language Models

6.1 Motivation

Rigor and in-depth interpretation are primary objectives in qualitative analysis (Watkins, 2017; Maher et al., 2018). Collaborative Qualitative Analysis (CQA) underscores this by mandating researchers to code individually and converge on interpretations through iterative discussions (Cornish, Gillespie, and Zittoun, 2013; Anderson, Guerreiro, and J. Smith, 2016; Hall et al., 2005; Richards and Hemphill, 2018; N. McDonald, Schoenebeck, and Forte, 2019) (see Figure 6.1). Such a method is instrumental in preserving rigor (Richards and Hemphill, 2018) and facilitating a richer, more nuanced grasp of data interpretation (Anderson, Guerreiro, and J. Smith, 2016).

However, adhering strictly to the CQA's prescribed workflow, which is integral for achieving both rigor and depth goals, poses challenges due to its *associated time and labor costs* as well as *inherent complexity*. For the former issue, a primary reason is that the iterative nature of CQA requires the involvement and coordination of many coders (Ganji, Orand, and D. W. McDonald, 2018; Gao, Choo, et al., 2023), but the conventional CQA tools such as MaxQDA, NVivo, and Google Docs/Sheets are not specifically designed for this aspect. They necessitate additional team coordination steps (Malone and Crowston, 1994; Entin, 2000), like document downloading, data sharing in the team, data importing, manual searching, and crafting codebook tables. For the latter issue, the complexity of the CQA process, which involves multiple steps with specific requirements for each, presents a considerable entry barrier for those less experienced or unfamiliar with CQA standards like graduate students, early-career researchers, and diverse research teams (Cornish, Gillespie, and Zittoun, 2013; Richards

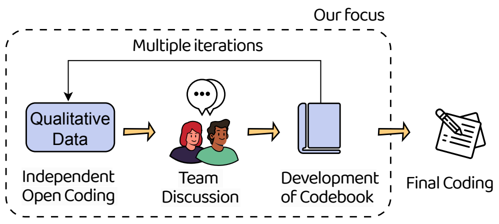


FIGURE 6.1: Collaborative Qualitative Analysis (CQA) (J. Corbin and Strauss, 2008; J. M. Corbin and Strauss, 1990; Richards and Hemphill, 2018) is an iterative process involving multiple rounds of iteration among coders to reach a final consensus. Our goal with CollabCoder is to assist users across key stages of the CQA process.

and Hemphill, 2018). For instance, Atlas.ti Web lacks an independent coding space. This absence means that the coding process is always visible and can potentially influence others' open coding (Gao, Choo, et al., 2023), which might lead to confusion or incorrect practices. Despite this challenge, most software is specifically engineered to support basic functions, such as proposing codes. They typically lack a comprehensive and holistic theoretical framework that could provide more effective assistance. This limitation leads to much confusion, even among those well-versed in CQA theories, compelling them to opt for independent coding methods in exchange for efficiency, resulting in fewer interactive discussions, diminished coding rigor and depth, and ultimately, the risk of the outcomes reflecting the individual coder's inherent biases (Cornish, Gillespie, and Zittoun, 2013; Anderson, Guerreiro, and J. Smith, 2016).

Current HCI researchers are mainly focusing on addressing effort-intensive challenges and have developed specialized tools to streamline various aspects of the CQA process. For example, Zade et al. (Zade et al., 2018) suggested enabling coders to order different states of disagreements by conceptualizing disagreements in terms of tree-based ranking metrics of diversity and divergence. Aeonium (Drouhard et al., 2017) allows coders to highlight ambiguity and inconsistency and offers features to navigate through and resolve them. With the growing prevalence of AI, Gao et al. (Gao, Choo, et al., 2023) underscores the potential of AI in CQA through CoAIcoder, suggesting that AI agents that provide code suggestions based on teams' coding histories accelerate collaborative efficiency and foster a shared understanding at the early stage of coding.

With the advancements of LLMs[1] like GPT-3.5 and GPT-4[2], they have been pivotal in enhancing qualitative analysis due to the exceptional abilities in understanding and generating text. Atlas.ti Web, a commercial platform for qualitative analysis, integrated

[1]In this paper, AI and LLMs are used interchangeably to refer to the broader field of Artificial Intelligence, specifically large language models. GPT, as an example of a large language model, specifically refers to products developed by OpenAI, such as ChatGPT.

[2]https://openai.com/blog/introducing-chatgpt-and-whisper-apis

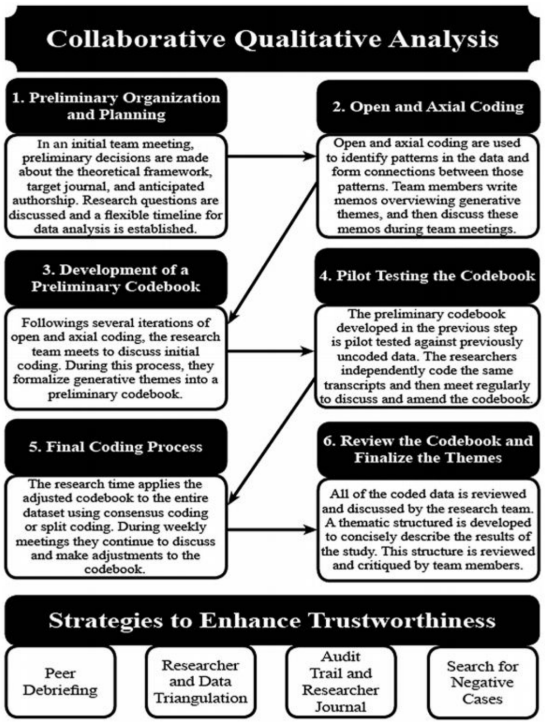

Collaborative Qualitative Analysis
1. Preliminary Organization and Planning
In an initial team meeting, preliminary decisions are made about the theoretical framework, target journal, and anticipated authorship. Research questions are discussed and a flexible timeline for data analysis is established.
2. Open and Axial Coding
Open and axial coding are used to identify patterns in the data and form connections between those patterns. Team members write memos overviewing generative themes, and then discuss these memos during team meetings.
3. Development of a Preliminary Codebook
Followings several iterations of open and axial coding, the research team meets to discuss initial coding. During this process, they formalize generative themes into a preliminary codebook.
4. Pilot Testing the Codebook
The preliminary codebook developed in the previous step is pilot tested against previously uncoded data. The researchers independently code the same transcripts and then meet regularly to discuss and amend the codebook.
5. Final Coding Process
The research time applies the adjusted codebook to the entire dataset using consensus coding or split coding. During weekly meetings they continue to discuss and make adjustments to the codebook.
6. Review the Codebook and Finalize the Themes
All of the coded data is reviewed and discussed by the research team. A thematic structured is developed to concisely describe the results of the study. This structure is reviewed and critiqued by team members.
Strategies to Enhance Trustworthiness
Peer Debriefing
Researcher and Data Triangulation
Audit Trail and Researcher Journal
Search for Negative Cases

FIGURE 6.2: Overview of the six steps involved in collaborative qualitative analysis proposed by Richards and Hemphill, 2018.

OpenAI's GPT model on March 28, 2023[3]. This integration offers functionalities like one-click code generation and AI-driven code suggestions, significantly streamlining the coding process. LLMs also assist in the deductive coding of large-scale datasets (Xiao et al., 2023) and are being explored for their potential to replace human coders in discerning subtle data nuances (Byun, Vasicek, and Seppi, 2023), among other applications. This integration serves as a key factor in reducing the labor intensity typically associated with traditional qualitative analysis.

While this existing research provides valuable insights into various facets of CQA, there has been little emphasis on creating a streamlined workflow to bolster the rigorous CQA process. Building upon well-accepted CQA steps that are deeply rooted in Grounded Theory (Clark and Brennan, 1991) and Thematic Analysis (Maguire and Delahunt, 2017), we aim to address this gap by presenting a holistic, one-stop solution that enhances the CQA process, with an emphasis on the *inductive* qualitative analysis and *consensus coding*, central to the development of the codebook and coding schema. This is in contrast to the work by Xiao et al. (Xiao et al., 2023), which prioritizes the use of LLMs to assist deductive coding based on a pre-existing codebook. The overarching aim is to lower the bar for maintaining the rigor and reliability of the inductive CQA

[3]https://atlasti.com/ai-coding-powered-by-openai

TABLE 6.1: Summary of Sources Informing Our Design Goals

Sources	Content	Design Goals (DG)
Step1: Semi-systematic literature review	insights into the key phases of CQA theories, including grounded theory and thematic analysis, detailing inputs, outputs, and practical considerations for each.	DG1, DG2, DG4, DG5, DG6, DG7
Step2: Examination of prevalent CQA platforms	insights into essential features, pros, and cons of key CQA platforms, such as Atlas.ti Web[4], MaxQDA Team Cloud[5], NVivo Collaboration Cloud[6], and Google Docs.	DG1, DG2, DG3, DG8
Step3: Preliminary interviews with researchers with qualitative analysis experience	insights into CollabCoder workflow, features, and design scope through expert feedback, concerns, and recommendations.	DG1, DG4

process. **Our primary objective is to lower the bar of adherence to the rigorous CQA process**, thereby providing a potential for enhancing the quality of qualitative interpretation (Collins and Stockton, 2018) with controllable and manageable effort.

6.2 Design Goals

6.2.1 Method

To achieve our goals, we extracted 8 design goals (DG) for CollabCoder from three primary sources (see Table 6.1).

Step1: Semi-systematic literature review We initially reviewed established theories and guidelines on qualitative analysis. Given our precise focus on theories such as Grounded Theory and Thematic Analysis and our emphasis on their particular steps, we used a semi-systematic literature review method (Snyder, 2019; Mäntylä et al., 2015). This method is particularly aimed at identifying key themes relevant to a specific topic while offering an appropriate balance of depth and flexibility. Our results are incorporated into the background section, aiming to establish a robust theoretical foundation for our work. It also assists in delineating the inputs, outputs, and practical considerations for each stage of CollabCoder workflow. This method formulates DG1, DG2, DG4, DG5, DG6, DG7.

Step2: Examination of prevalent CQA platforms The semi-systematic literature review was followed by triangulation with existing qualitative analysis platforms, for which we assessed the current state of design by examining their public documents and official websites (the detailed examination are summarized in Appendix Table B.1). This examination enables us to gain insights into the critical features, advantages, and drawbacks of these CQA platforms, such as the dropdown list for the selection of historical codes and the calculation of essential analysis metrics. As a result of this triangulation, we successfully extracted new design goals, DG3 and DG8, and refined the existing DG1 and DG2.

Step3: Preliminary interviews with researchers with qualitative analysis experience Based on the primary understanding of the CQA theories, and the primary version of 8 DGs, we subsequently developed the primary prototype (see Appendix Figures B.1, B.2, and B.3). We utilized the initial version of CollabCoder to conduct a pilot interview evaluation with five researchers possessing at least one year of experience in qualitative analysis (refer to Table 6.2). The aim was to gather expert insights into the workflow, features, and design scope of the theory-driven CollabCoder, thereby refining our design goals and adjusting the prototype's primary features. During the evaluation, the researchers were first introduced to the CollabCoder prototype. Subsequently, they shared their impressions, raised questions, and offered suggestions for enhancements. We transcribed their interview audio and did a thematic analysis on the interview transcriptions (see analysis results in Appendix Figure B.4) and refined two of the design goals (DG1 and DG4) based on their feedback.

TABLE 6.2: Participant Demographics in Exploration Interview

No.	Fields of Study	Current Position	QA Software	Years of QA
P1	HCI, Ubicomp	Postdoc Researcher	Atlas.ti	4.5
P2	HCI, NLP	PhD student	Google Sheet/ Whiteboard	4
P3	HCI, Health	PhD student	Google Sheet	4
P4	HCI, NLP	PhD student	Excel	1.5
P5	Software Engineering	PhD student	Google Sheet	1

6.2.2 Results for Design Goals

DG1: Supporting key CQA phases to encourage stricter adherence to standardized CQA processes Our primary goal is the creation of a mutually agreed codebook among coders, essentially focusing on the inductive qualitative analysis process. Therefore, from the six-step CQA methodology (Richards and Hemphill, 2018), we are particularly concerned with "open and axial coding", "iterative discussion", and the "development of a codebook".

Although complying with CQA steps is critical for deriving robust and trustworthy data interpretations (Richards and Hemphill, 2018), the existing software workflows and AI integrations are quite demanding. These systems currently do not offer a centralized and focused workflow; there is a noticeable absence of fluidity between stages, where the output of one phase should ideally transition seamlessly into the input of the next. This deficiency complicates the sensemaking process among coders and often discourages them from adhering to the standardized CQA workflow. This sentiment is mirrored by an expert (P1) who remarked, *"In a realistic scenario, how many people do follow this [standard] flow? I don't think most people follow."*"

In response, we have tailored a workflow that integrates the key CQA stages we identified. This streamlined process assists the coding team in aligning with the standard coding procedure, ensuring results from one phase transition seamlessly into the next. Our goal is to simplify adherence to the standard workflow, making it more accessible.

DG2: Supporting varying levels of coding independence at each CQA stage to ensure a strict workflow. In Grounded Theory (J. Corbin and Strauss, 2008; Richards and Hemphill, 2018), a primary principle is to enable coders to independently produce codes, cultivate insights from their own viewpoints, and subsequently share these perspectives at later stages. However, we have found that widely-used platforms such as Atlas.ti Web and NVivo, while boasting real-time collaborative coding features, fall short in providing robust support for independent coding. The persistent visibility of all raw data, codes, and quotations to all participants may potentially bias the coding process. Moreover, Gao et al. (Gao, Choo, et al., 2023) also found that in scenarios prioritizing efficiency, some coders are willing to compromise independence, which could potentially impact coding rigor.

In response, our workflow designates varying levels of coder independence at different stages: strict separation during the independent open coding phase and mutual code access in discussion and code grouping phases. We aim to ensure that coders propose codes from their unique perspectives, rather than prematurely integrating others' viewpoints, which could compromise the final coding quality.

DG3: Supporting streamlined data management and synchronization within the coding team. While Atlas.ti Web has faced criticism for its lack of support for coder independence (Gao, Choo, et al., 2023), as outlined in DG2, it does offer features like **synchronization** and **centralized data management**. Through a web-based application, these features allow teams to manage data preprocessing and share projects. This ensures seamless coding synchronization among members. The sole requirement for participation is a web browser and an Atlas.ti Web account. In contrast, traditional software like MaxQDA and nVivo lack these capabilities. This absence necessitates additional steps, such as locally exporting coding documents post-independent coding and then sharing them with team members[7]. These steps may introduce obstacles to a smooth and focused CQA process. However, as mentioned in DG2, Atlas.ti Web sacrifices coding independence.

[7]https://www.maxqda.com/help-mx20/teamwork/can-maxqda-support-teamwork

In response, we strive to strike a balance between data management convenience and coding independence, facilitating seamless data synchronization and management via a web application while maintaining design features that support independent coding.

DG4: Supporting interpretation at the same level among coders for efficient discussion. As per Saldana's qualitative coding manual (Saldaña, 2021), coders may use a "splitter" (e.g., line-by-line) or a "lumper" (e.g., paragraph-by-paragraph) approach. This variation can lead coders to work on different levels of granularity, resulting in many extra efforts to align coding units among coders for line-by-line or code-by-code comparison, in order to make them on the same level to determine if they have an agreement or consensus, not to mention the calculation of IRR (Gao, Choo, et al., 2023). Therefore, standardizing and aligning data units for coding among teams is essential to facilitate efficient code comparisons and IRR calculations (Kurasaki, 2000; O'Connor and Joffe, 2020; Ganji, Orand, and D. W. McDonald, 2018). Two prevalent approaches to achieve this are: 1) allowing the initial coder to finish coding before another coder commences work on the same unit (J. Díaz et al., 2023; Kurasaki, 2000; O'Connor and Joffe, 2020), and 2) predefining a fixed text unit for the team, such as sentences, paragraphs, or conceptually significant "chunks" (O'Connor and Joffe, 2020; Kurasaki, 2000).

In response, we aim to enhance code comparison efficiency by offering coders predefined coding unit options on CollabCoder, thereby ensuring alignment between their interpretations. However, it is important to recognize an intrinsic trade-off between **unit selection flexibility** and **effort expenditure**. While reduced flexibility can decrease the effort needed to synchronize coders' understanding in discussions, it may also constrain users' freedom in coding. According to expert feedback, our workflow represents an "ideal" scenario. As one expert (P3) noted, *"I think overall the CollabCoder workflow pretty interesting... However, I think the current workflow is a very perfect scenario. What you haven't considered is that in qualitative coding, there's often a sentence or a section that can be assigned to multiple codes. In your current case, you are assigning an entire section into just one code."* Additionally, our proposed workflow appears to operate under the assumption that coding is applied to specific, isolated units, failing to account for instances where the same meaning is distributed across different data segments. *"Because sometimes [for a code] you need one part of one paragraph, the other part is in another paragraph. right?"* (P1)

DG5: Supporting coding assistance with LLMs while preserving user autonomy. As Jiang et al. (J. A. Jiang et al., 2021) suggested, AI should not replace human autonomy, participants in their interview said that *"I don't want AI to pick the good quotes for me..."*. AI should only offer recommendations when requested by the user, after they have manually labeled some codes, and support the identification of overlooked codes based on existing ones. To control user autonomy, the commercial software, Atlas.ti Web, has transitioned from auto-highlighting quotations and generating code suggestions via LLMs for all documents with a single click, to now allowing users to request such suggestions on demand [8]. The platform's earlier AI-driven coding, although time-saving, compromised user control in the coding process.

In response, we emphasize user autonomy during the coding process, letting coders first formulate their own codes and turning to LLM assistance only upon request.

DG6: Facilitating deeper and higher-quality discussion. CollabCoder's primary objective is to foster consensus among coders (Anderson, Guerreiro, and J. Smith, 2016; Richards and Hemphill, 2018). This demands quality discussions rooted in common ground. Common ground (Patel, Pettitt, and Wilson, 2012; G. M. Olson and J. S. Olson, 2000) pertains to the information that individuals have in common and are aware that others possess, a notion rooted in the grounding process in communication (Clark and Brennan, 1991; Bjørn et al., 2014). Grounding is achieved when collaborators engage in deep communication (Bjørn et al., 2014). A lack of common ground can lead to distrust, misunderstandings, poor team performance and decision-making.

In response, we aim to establish common ground between coders, in order to: 1) facilitate deeper and higher-quality discussion by surfacing underlying coding disagreements; 2) concentrate coders' efforts on the most critical parts that need the most discussion (Drouhard et al., 2017; Zade et al., 2018).

DG7: Facilitating cost-effective, fair coding outcomes and engagement via LLMs. Once the common ground is established, achieving a coding outcome that is cost-effective, fair, and free from negative effects becomes a challenging yet crucial task (Jameson, Baldes, and Kleinbauer, 2003; Emamgholizadeh, 2022). To reach a consensus, the team often engages in debates or invests time crafting code expressions that satisfy all coders (Emamgholizadeh, 2022), significantly prolonging the discussion. In addition, Jiang et al. (J. A. Jiang et al., 2021) reveal that team leaders or senior members may have the final say on the codes, potentially introducing bias.

In response, our objective is to foster deep, efficient, and balanced discussions within the coding team. We ensure that every coder's prior open coding decisions are respected, allowing them to actively participate in both discussions and the final decision-making process, with the support of LLMs.

DG8: Enhancing the team's efficiency in code group generation Prevalent QA software like Atlas.ti, MaxQDA, and nVivo prominently feature a code manager. This tool

[8]`https://atlasti.com/atlas-ti-ai-lab-accelerating-innovation-for-data-analysis`, accessed on 14th August 2023

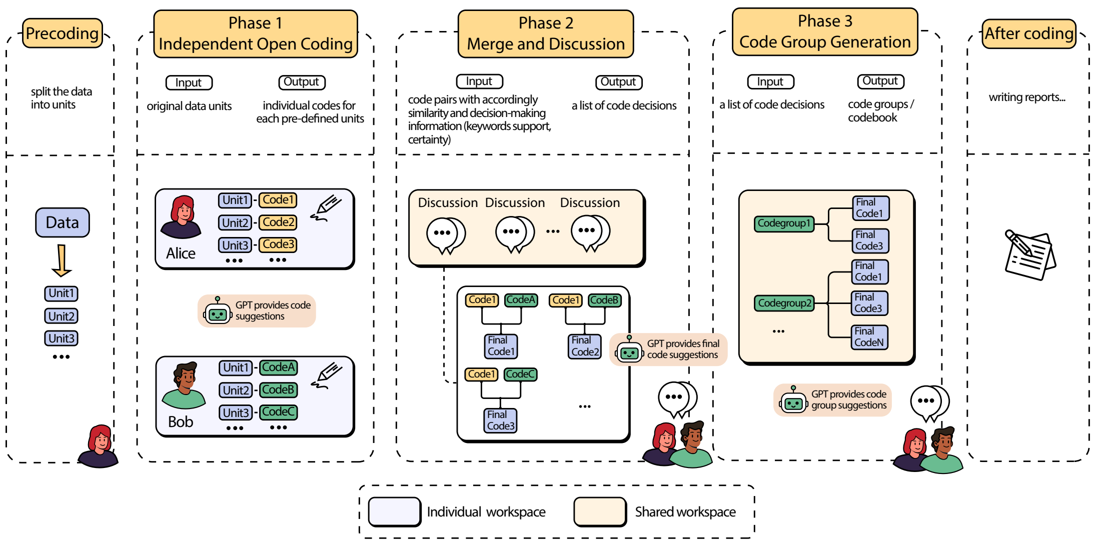

Precoding
split the data into units
Data
Unit1
Unit2
Unit3
Phase 1 Independent Open Coding
Input
original data units
Output
individual codes for each pre-defined units
Alice
Bob
GPT provides code suggestions
Phase 2 Merge and Discussion
Input
code pairs with accordingly similarity and decision-making information (keywords support, certainty)
Output
a list of code decisions
Discussion
GPT provides final code suggestions
Phase 3 Code Group Generation
Input
a list of code decisions
Output
code groups / codebook
Codegroup1
Codegroup2
GPT provides code group suggestions
After coding
writing reports...
Individual workspace
Shared workspace

FIGURE 6.3: CollabCoder Workflow. The lead coder Alice first splits qualitative data into small units of analysis, e.g., sentence, paragraph, prior to the formal coding. Alice and Bob then: Phase 1: independently perform open coding with GPT assistance; Phase 2: merge, discuss, and make decisions on codes, assisted by GPT; Phase 3: utilize GPT to generate code groups for decided codes and perform editing. They can write reports based on the codebook and the identified themes after the formal coding process.

lets coders track, modify, and get a holistic view of their current code assignments. It plays a vital role in facilitating discussions, proposing multiple code groups, and aiding code reuse during coding. Meanwhile, Feuston et al. (Feuston and Brubaker, 2021) noted some participants used AI tools to auto-generate final code groups from human-assigned codes.

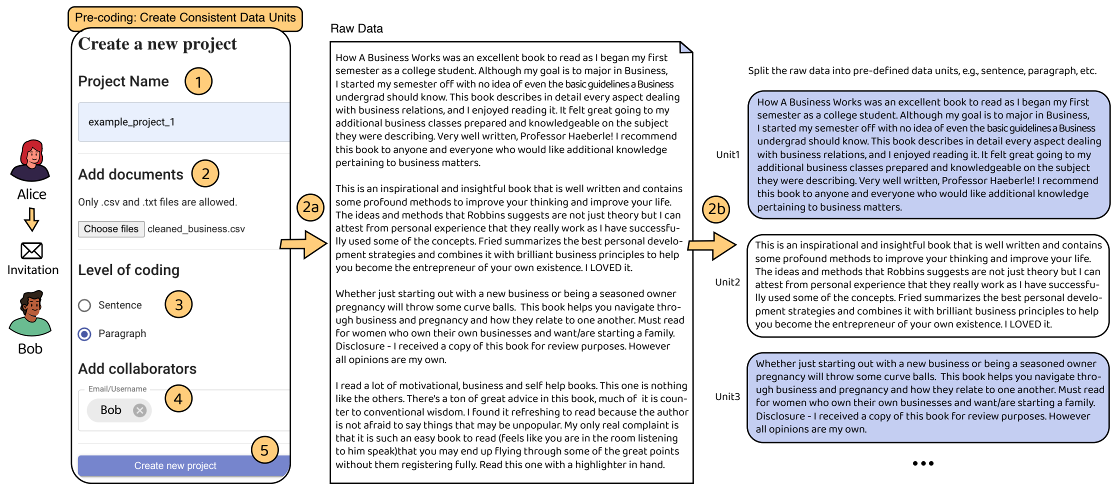
In response, we offer the code manager that allows for manual editing and adjustment of code groups. Additionally, we aim to integrate automatic code group generation to streamline the coding process via the assistance of LLMs.

6.3 CollabCoder System

With the aforementioned design goals in mind, we have finalized the CollabCoder system and its CQA workflow (refer to Figure 6.3).

6.3.1 CollabCoder Workflow & Usage Scenario

We introduce an example scenario to demonstrate the usage of CollabCoder (see Figure 6.5). Suppose two coders **Alice** and **Bob** are conducting qualitative coding for their data. The lead coder, **Alice**, first creates a new project on CollabCoder, then imports



Create a new project

Project Name: example_project_1

Add documents
Only .csv and .txt files are allowed.
Choose files cleaned_business.csv

Level of coding
( ) Sentence
(●) Paragraph

Add collaborators
Email/Username: Bob

Create new project

Raw Data

How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters.

This is an inspirational and insightful book that is well written and contains some profound methods to improve your thinking and improve your life. The ideas and methods that Robbins suggests are not just theory but I can attest from personal experience that they really work as I have successfully used some of the concepts. Fried summarizes the best personal development strategies and combines it with brilliant business principles to help you become the entrepreneur of your own existence. I LOVED it.

Whether just starting out with a new business or being a seasoned owner pregnancy will throw some curve balls. This book helps you navigate through business and pregnancy and how they relate to one another. Must read for women who own their own businesses and want/are starting a family. Disclosure - I received a copy of this book for review purposes. However all opinions are my own.

I read a lot of motivational, business and self help books. This one is nothing like the others. There's a ton of great advice in this book, much of it is counter to conventional wisdom. I found it refreshing to read because the author is not afraid to say things that may be unpopular. My only real complaint is that it is such an easy book to read (feels like you are in the room listening to him speak)that you may end up flying through some of the great points without them registering fully. Read this one with a highlighter in hand.

Split the raw data into pre-defined data units, e.g., sentence, paragraph, etc.

Unit1: How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters.

Unit2: This is an inspirational and insightful book that is well written and contains some profound methods to improve your thinking and improve your life. The ideas and methods that Robbins suggests are not just theory but I can attest from personal experience that they really work as I have successfully used some of the concepts. Fried summarizes the best personal development strategies and combines it with brilliant business principles to help you become the entrepreneur of your own existence. I LOVED it.

Unit3: Whether just starting out with a new business or being a seasoned owner pregnancy will throw some curve balls. This book helps you navigate through business and pregnancy and how they relate to one another. Must read for women who own their own businesses and want/are starting a family. Disclosure - I received a copy of this book for review purposes. However all opinions are my own.

FIGURE 6.4: Precoding: establish consistent data units and enlist coding team during project creation. The primary coder, Alice, can: 1) name the project, 2) incorporate data, ensuring it aligns with mutually agreed data units, 2a) illustrate how CollabCoder manages the imported data units, 3) define the coding granularity (e.g., sentence or paragraph), 4) invite a secondary coder, Bob, to the project, and 5) initiate the project.

the data, specifies the level of coding as "paragraph", and invites **Bob** to join the project. After clicking on CREATE PROJECT, CollabCoder's parser will split the imported raw data into units (paragraph in this case). The project can then be shown on both coders' interfaces.

Phase 1: Independent Open Coding

In Phase 1, **Alice** and **Bob** individually formulate codes for each unit in their separate workspaces via the same interface. Their work is done independently, with no visibility into each other's codes. If **Alice** wants to propose a code for a sentence describing a business book for students, she can either *craft* her own code, choose from code recommendations generated by the GPT model (e.g., `"Excellent guide for new college students"`,`"Insightful read on business fundamentals"`,`"How A Business Works": semester's gem`), or *pick* one of the top three most relevant codes discovered by GPT in her coding history, and making modifications as needed. She can then *select* relevant keywords/phrases (e.g., `"excellent book"`, `"college student"`) from the RAW DATA cell that supports her proposed code, which will be added to the KEYWORDS SUPPORT beside her proposed code. She can also *assign* a CERTAINTY, ranging from 1 to 5, to the code. This newly generated code will be included in **Alice**'s personal CODEBOOK and can be *viewed* by her at any time. They can *check* the progress of each other in the PROGRESS at any time (see Figure 6.6).

Phase 2: Code Merging and Discussion

Figure 6.6 depicts the shared workspace where coding teams collaborate, discussing their code choices and making final decisions regarding the codes identified in Phase 1. After completing coding, **Alice** can *check* the CHECKBOX next to **Bob**'s name once she sees that his progress is at 100%. Subsequently, she can *click* the CALCULATE button to generate quantitative metrics such as similarity scores and IRR (Cohen's Kappa and Agreement Rate[9]) within the team. The rows are then sorted by similarity scores in descending order.

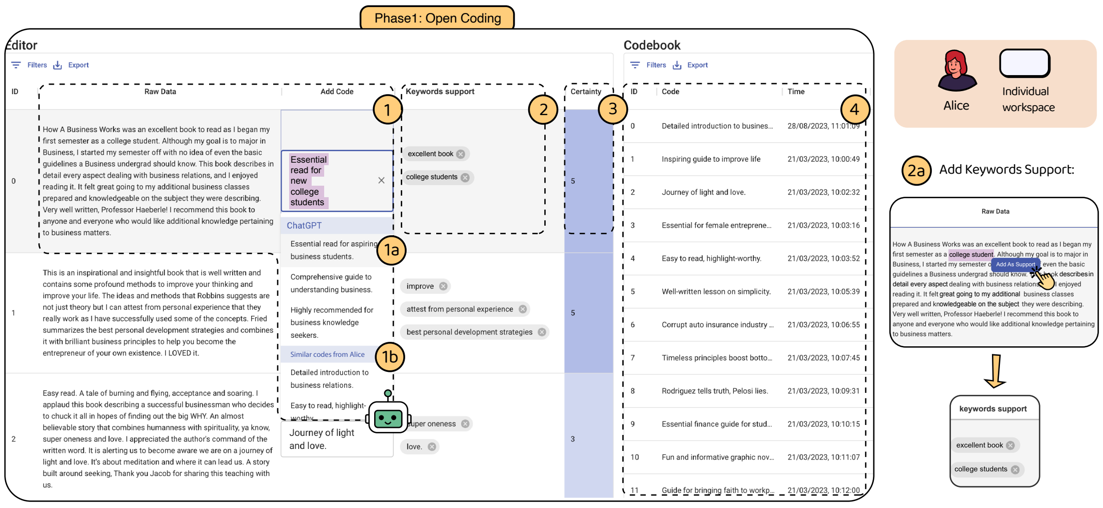
Alice can then share her screen via a Zoom meeting with **Bob** to COMPARE AND DISCUSS their codes, starting from code pairs with high similarity scores. For instance, **Alice**'s code `"Excellent guide for new college students"` with a certainty of 5 includes `"excellent book"` and `"college student"` supports, while **Bob**'s code `"Excellent read for aspiring business students"` with a certainty of 4 includes `"How A business works"` and `"as a college student"` as KEYWORDS SUPPORT. The similarity score between their codes could be 0.876 (close to 1), showing a high agreement. During the discussion, they both agreed that the final code should contain the word "student" due to their similar KEYWORDS SUPPORT, but they cannot reach a consensus about the final expression of the code, they then seek GPT suggestions (e.g., `"Essential college guide for business students"`, `"Semester's gem for new college students"`,`Essential college starter`), and decide the final code decision for this unit is `"Essential college guide for business students"`. However, if the code pair presents a low similarity score, they must allocate additional time to scrutinize the code decision information and identify the keywords that led to different interpretations.

Once all code decisions have been made, **Alice** can then *click* on REPLACE to replace the original codes, resulting in an update of Cohen's Kappa and Agreement Rate. This action can be undone by clicking on UNDO.

Phase 3: Code Group Generation

Once **Alice** and **Bob** have agreed on the final code decisions for all the units, the code decision list will be displayed on the code grouping interface, as shown in Figure 6.7. This interface is shared uniformly among the coding team. For further discussion, **Alice** can continue to share her screen with **Bob** on Zoom. She can *hover over* each CODE DECISION to refer to the corresponding raw data or *double-click* to edit. **Alice** and **Bob** can collaborate to propose the final code groups by *clicking* on ADD NEW

[9]The calculation methods differ between these two metrics. Cohen's kappa is a more intricate method for measuring agreement, as detailed in McHugh, 2012. On the other hand, the Agreement Rate represents the percentage of data on which coders concur.



ID	Raw Data	Add Code	Keywords support	Certainty
0	How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters.	Essential read for new college students	excellent book; college students	5
1	This is an inspirational and insightful book that is well written and contains some profound methods to improve your thinking and improve your life. The ideas and methods that Robbins suggests are not just theory but I can attest from personal experience that they really work as I have successfully used some of the concepts. Fried summarizes the best personal development strategies and combines it with brilliant business principles to help you become the entrepreneur of your own existence. I LOVED it.		improve; attest from personal experience; best personal development strategies	5
2	Easy read. A tale of burning and flying, acceptance and soaring. I applaud this book describing a successful businessman who decides to chuck it all in hopes of finding out the big WHY. An almost believable story that combines humanness with spirituality, ya know, super oneness and love. I appreciated the author's command of the written word. It is alerting us to become aware we are on a journey of light and love. It's about meditation and where it can lead us. A story built around seeking, Thank you Jacob for sharing this teaching with us.	Journey of light and love.	super oneness; love.	3

ChatGPT:
- Essential read for aspiring business students.
- Comprehensive guide to understanding business.
- Highly recommended for business knowledge seekers.

Similar codes from Alice:
- Detailed introduction to business relations.
- Easy to read, highlight-worthy.

Codebook

ID	Code	Time
0	Detailed introduction to busines...	28/08/2023, 11:01:09
1	Inspiring guide to improve life	21/03/2023, 10:00:49
2	Journey of light and love.	21/03/2023, 10:02:32
3	Essential for female entreprene...	21/03/2023, 10:03:16
4	Easy to read, highlight-worthy.	21/03/2023, 10:03:52
5	Well-written lesson on simplicity.	21/03/2023, 10:05:39
6	Corrupt auto insurance industry ...	21/03/2023, 10:06:55
7	Timeless principles boost botto...	21/03/2023, 10:07:45
8	Rodriguez tells truth, Pelosi lies.	21/03/2023, 10:09:31
9	Essential finance guide for stud...	21/03/2023, 10:10:15
10	Fun and informative graphic nov...	21/03/2023, 10:11:07
11	Guide for bringing faith to workp...	21/03/2023, 10:12:00

Raw Data
How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester o even the basic guidelines a Business undergrad should know. ok describes in detail every aspect dealing with business relations I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters.

keywords support
excellent book
college students

FIGURE 6.5: Editing Interface for Phase 1: 1) inputting customized code for the text in "Raw Data", either 1a) choosing from the GPT's recommendations, 1b) choosing from the top three relevant codes; 2) adding keywords support by 2a) selecting from raw data and "Add As Support"; 3) assigning a certainty level ranging from 1 to 5, where 1="very uncertain" and 5="very certain"; and 4) reviewing and modifying the individual codebook.

GROUP and *drag* the code decisions into the new code group. For instance, a group "`Business knowledge`" can include "`Simplified business knowledge`", "`Cautionary book on costly Google campaigns`" and others. Alternatively, they can request GPT assistance by *clicking* on the CREATE CODE GROUPS BY AI button to automatically generate several code groups and place the individual code decisions into them. These groups can still be manually adjusted by coders. Once they finish grouping, they can proceed to report their findings as necessary.

6.3.2 Key Features

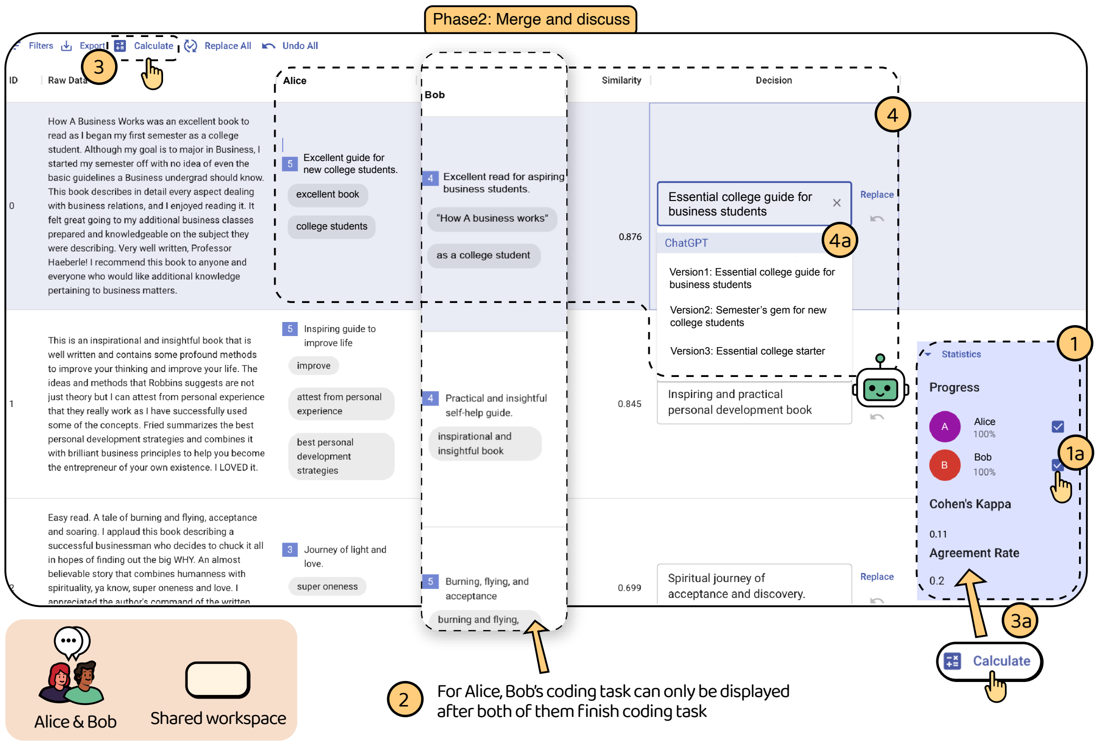
Three-phase Interfaces

In alignment with DG1, our objective was to incorporate a workflow that supports the three key phases of the CQA process, as derived from established theories. Accordingly, our system is segmented into three distinct interfaces:

1. Editing Interface for Phase 1: Independent Open Coding (Figure 6.5).

2. Comparison Interface for Phase 2: Merge and Discuss (Figure 6.6).

3. Code Group Interface for Phase 3: Code Groups Generation (Figure 6.7).

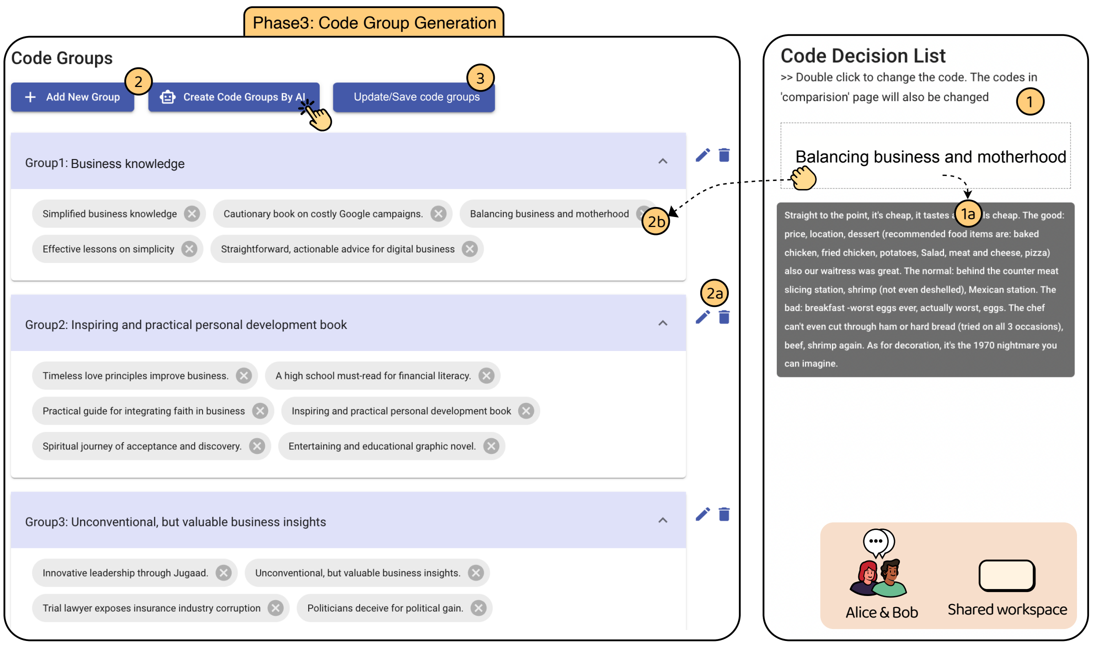


ID	Raw Data	Alice	Bob	Similarity	Decision
0	How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters.	5 Excellent guide for new college students. / excellent book / college students	4 Excellent read for aspiring business students. / "How A business works" / as a college student	0.876	Essential college guide for business students / ChatGPT / Version1: Essential college guide for business students / Version2: Semester's gem for new college students / Version3: Essential college starter
1	This is an inspirational and insightful book that is well written and contains some profound methods to improve your thinking and improve your life. The ideas and methods that Robbins suggests are not just theory but I can attest from personal experience that they really work as I have successfully used some of the concepts. Fried summarizes the best personal development strategies and combines it with brilliant business principles to help you become the entrepreneur of your own existence. I LOVED it.	5 Inspiring guide to improve life / improve / attest from personal experience / best personal development strategies	4 Practical and insightful self-help guide. / inspirational and insightful book	0.845	Inspiring and practical personal development book
2	Easy read. A tale of burning and flying, acceptance and soaring. I applaud this book describing a successful businessman who decides to chuck it all in hopes of finding out the big WHY. An almost believable story that combines humanness with spirituality, ya know, super oneness and love. I appreciated the author's command of the written	3 Journey of light and love. / super oneness	5 Burning, flying, and acceptance / burning and flying,	0.699	Spiritual journey of acceptance and discovery.

FIGURE 6.6: Comparison Interface for Phase 2. Users can discuss and reach a consensus by following these steps: 1) reviewing another coder's progress and 1a) clicking on the checkbox **only if** both individuals complete their coding, 2) two coders' codes are listed in the same interface, 3) calculating the similarity between code pairs and 3a) IRR between coders, 4) sorting the similarity scores from highest to lowest and identifying (dis)agreements, and 4a) making a decision through discussion based on the initial codes, raw data, and code supports or utilizing the GPT's three potential code decision suggestions. Additionally, users have the option to "Replace" the original codes proposed by two coders and revert back to the original codes if required. They can also replace or revert all code decisions with a single click on the top bar.

Individual Workspace vs. Shared Workspace

Aligned with DG2, we aim to mirror the distinct levels of independence intrinsic to the CQA process, reflecting the principles of qualitative analysis theories. CollabCoder introduces an *"individual workspace"* — the Editing Interface — allowing users to code individually during the initial phase without visibility of others' coding. Additionally, for facilitating Phase 2 discussions, CollabCoder unveils a "shared workspace." Here, the checkbox next to each coder's name activates only after both participants complete their individual coding, represented as percentages (0-100%). This shared interface enables the team to collectively review and discuss coding data within an integrated environment.



Group1: Business knowledge

Simplified business knowledge · Cautionary book on costly Google campaigns. · Balancing business and motherhood · Effective lessons on simplicity · Straightforward, actionable advice for digital business

Group2: Inspiring and practical personal development book

Timeless love principles improve business. · A high school must-read for financial literacy. · Practical guide for integrating faith in business · Inspiring and practical personal development book · Spiritual journey of acceptance and discovery. · Entertaining and educational graphic novel.

Group3: Unconventional, but valuable business insights

Innovative leadership through Jugaad. · Unconventional, but valuable business insights. · Trial lawyer exposes insurance industry corruption · Politicians deceive for political gain.

>> Double click to change the code. The codes in 'comparision' page will also be changed

Balancing business and motherhood

Straight to the point, it's cheap, it tastes a[illegible]ls cheap. The good: price, location, dessert (recommended food items are: baked chicken, fried chicken, potatoes, Salad, meat and cheese, pizza) also our waitress was great. The normal: behind the counter meat slicing station, shrimp (not even deshelled), Mexican station. The bad: breakfast -worst eggs ever, actually worst, eggs. The chef can't even cut through ham or hard bread (tried on all 3 occasions), beef, shrimp again. As for decoration, it's the 1970 nightmare you can imagine.

FIGURE 6.7: Code Group Interface. It enables users to manage their code decisions in a few steps: 1) the code decisions are automatically compiled into a list of unique codes that users can edit by double-clicking and accessing the original data by hovering over the code. 2) users can group their code decisions by using either "Add New Group" or "Create Code Groups By AI" options. They can then 2a) name or delete a code group or use AI-generated themes, and 2b) drag the code decisions into code groups. 3) Finally, users can save and update the code groups.

Web-based Platform

In alignment with DG3, our goal is to harness the synchronization benefits of Atlas.ti Web while preserving the essential independence required for the CQA process. CollabCoder addresses this by using a web-based platform. Here, the lead coder creates a project and invites collaborators to engage with the same project. As outlined in section 6.3.2, upon the completion of individual coding, participants can effortlessly view the results of others, eliminating the need for downloads, imports, or further steps.

Consistent Data Units for All Users

Aligned with DG4, our objective is to synchronize coders' interpretation levels to boost discussion efficiency. CollabCoder facilitates this by segmenting data into uniform units (e.g., sentences or paragraphs) that are collaboratively determined by all coders prior to data importation or the onset of coding task.

LLMs-generated Coding Suggestions Once the User Requests

Aligned with DG5, we aim to empower coders to initially develop their own codes and then seek LLMs' assistance when necessary, striking a balance between user autonomy and the advantages of LLMs' support. Apart from proposing their own codes by themselves, CollabCoder offers LLMs-generated code suggestions when a user interacts with the input cell. These suggestions appear in a dropdown list for the chosen data unit after a brief delay, allowing users time to think about their own codes first. At the same time, CollabCoder identifies and provides the three most relevant codes from the current individual codebook for the given text unit, ensuring coding consistency when reusing established codes.

A Shared Workspace for Deeper Discussion

In alignment with DG6, our goal is to establish a shared understanding and foster richer, more substantive discussions. CollabCoder supports this goal through three key features.

1. *Documenting Decision-making Rationale.* In Phase 1, CollabCoder allows users to select keywords, phrases, and their coding certainty as supporting evidence. These highlighted elements can represent pivotal factors influencing the user's coding decision. CollabCoder further facilitates users in rating their certainty for each code on a scale from 1 (least certain) to 5 (most certain) to mark the ambiguity.

2. *Side-by-Side Comparison in A Shared Workspace*. Building on DG6's emphasis on establishing common ground, CollabCoder presents all users' coding information for the relevant

data units side-by-side. This display includes the original data units, supporting keywords, and indicators of labeled certainty scores. This layout facilitates direct comparison and nuanced discussions.

3. *Identifying (Dis)agreements.* CollabCoder simplifies the process of spotting (dis)agreements by calculating the Similarity of the code pair of each unit. This analysis can be executed in 3-10 seconds for all data units. Similarity scores for code pairs range from 0 (low similarity) to 1 (high similarity). For ease of interpretation, these scores can be sorted in descending order, with higher scores indicating stronger agreements.

LLMs as a Group Recommender System

In alignment with DG7, our aim is to foster cost-effective and equitable coding outcomes utilizing LLMs. CollabCoder achieves this by serving as an LLM-based group recommender system (Jameson, Baldes, and Kleinbauer, 2003): when users struggle to finalize a code, CollabCoder proposes three code decision suggestions specific to the code pair, taking into account the raw data, codes from each user, keywords support, and certainty scores. Users can then select and customize these suggestions to reach a conclusive coding decision.

Formation of LLMs-based Code Groups

Consistent with DG8, our objective is to optimize the process of code group creation to enhance efficiency. To this end, CollabCoder introduces the Code Group interface to provide two key functions:

1. *Accessing Original Data via the Final Code Decision List.* CollabCoder streamlines final code decisions, presenting them on the right-hand side of the interface. Hovering over a code reveals its originating raw data. Additionally, by double-clicking on an item within the code decision list, users can amend it, and the corresponding codes are updated accordingly.

2. *Managing Code Groups.* With CollabCoder, users can effortlessly craft, rename, or delete code groups. They can drag codes from the decision list to a designated code group or remove them. To save users the effort of building groups from scratch, CollabCoder provides an option to enlist GPT's help in organizing code decisions into preliminary groupings. This offers a foundation that users can then adjust, rename, or modify.

6.3.3 Prompts Design

CollabCoder leverages OpenAI's ChatGPT model (gpt-3.5-turbo) [10] to provide code and code group suggestions. Prompts we used are described in the following and all prompts template are also listed in Appendix Table B.2, B.3 and B.4. To ensure code suggestions have diversity without being overly random, the temperature parameter is set at 0.7.

Phase 1: Code Suggestions Recommendation

To recommend code suggestions, GPT is asked to play the role of:

"A helpful qualitative analysis assistant, aiding researchers in developing codes that can be utilized in subsequent stages, including discussions for creating codebooks and final coding processes".

CollabCoder recommends two kinds of code suggestions by prompting GPT:

① **Descriptive codes for raw data.**

Please create three general summaries for [text] (within six-words).

The six-word constraint was introduced after observing GPT's tendency to generate long summaries during testing. This limitation ensures that GPT delivers concise and targeted code suggestions. By using "general", we want GPT to generate codes that are not too specific to be reused.

② **Relevant codes derived from coding history.** CollabCoder produces the three most relevant codes from coding history using the following prompt:

Identify the top three codes relevant to this [text] from the following code list:
1. [Code]
2. [Code]...
Here is the format of the returned results:
1. code content
2. code content
3. code content.

Upon a user's request, two prompts, original text and code list (specified in *[text]* and *[Code]*) are sent to the OpenAI API, which generates three distinct code suggestions, and the three most relevant codes from coding history. The responses for both parts appear in a dropdown list when the user clicks on the editing cell, allowing for easy selection.

Phase 2: Code Decisions Recommendation

To generate code decisions upon user request, we ask GPT to play the role of:

[10] https://platform.openai.com/docs/models/gpt-3-5

"You are a helpful qualitative analysis assistant, aiding researchers in developing final codes that can be utilized in subsequent stages, including final coding processes."

We then prompt GPT with:

Create three concise, non-repetitive, and general six-word code combinations for the [text] using Code1 ([Code]) and Code2 ([Code])

Code1 and Code2 represent codes from Coder1 and Coder2 for the given raw data unit ([text]).

To ensure GPT provides results in a consistent format, we added the format and content requirements for the return results:

Here is the format of results:
Version1:
Version2:
Version3:

Requirements:
1. 6 words or fewer;
2. No duplicate words;
3. Be general;
4. Three distinct versions.

Phase3: Code Groups Recommendation

To facilitate the creation of primary groups, we ask GPT to play the role of:

"You are a helpful qualitative analysis assistant, aiding researchers in generating final code groups/main themes based on the [Code list] provided, in order to give an overview of the main content of the coding."

We prompt GPT with:

Organize the following codes into several thematic groups without altering the original codes, and name each group:
1. [Code],
2. [Code]...

Here is the format of the results:
Group1: [theme],
1.[code],
2.[code],
3.[code].
...

6.3.4 System Implementation

Web Application

The front-end implementation makes use of the react-mui library[11]. Specifically, we employed the DataGrid component[12] to construct tables in both the "Edit" and "Compare" interfaces, allowing users to input and compare codes. These tables auto-save user changes through HTTP requests to the backend, storing data in the database to synchronize progress among collaborators. For each data unit, users have their own code, keyword supports, certainty levels, and codebook in the Edit interface, while sharing decisions in the "Compare" interface and code groups in the "Codebook" interface. To prevent users from viewing collaborators' codes before editing is complete, we restrict access to other coders' codes and only show everyone's progress in "Compare" interface. We also utilized the foldable Accordion component[13] to efficiently display code group lists in the "Codebook" interface, where users can edit, drag and drop decision objects to modify their code groups. The backend leverages the Express framework, facilitating communication between the frontend and MongoDB. It also manages API calls to the GPT-3.5 model and uses Python to calculate statistics such as similarities.

Data Pre-processing

We partitioned raw data from CSV and txt files into data units during the pre-processing phase. At the sentence level, we segmented the text using common sentence delimiters such as ".", "...", "!", and "?". At the paragraph level, we split the text using `\n\n`.

Semantic Similarity and IRR

In CollabCoder, the IRR is measured using Cohen's Kappa[14] and Agreement Rate. To calculate Cohen's Kappa, we used the "cohen_kappa_score" method from scikit-learn package backend[15]. Cohen's Kappa is a score between -1 (total disagreement) and +1 (total agreement). Subsequently, we calculate the Agreement Rate as a score between 0 and 1, by determining the percentage of code pairs whose similarity score exceeds 0.8, indicating that the two coders agree on the code segment. We employ the *sentence-transformers* package[16] to determine the semantic similarity between pairs of code from two coders.

[11] https://mui.com/
[12] https://mui.com/x/react-data-grid/
[13] https://mui.com/material-ui/react-accordion/
[14] Cohen's Kappa is a statistical measure used to evaluate the IRR between two or more raters, which takes into account the possibility of agreement occurring by chance, thus providing a more accurate representation of agreement than simply calculating the percentage of agreement between the raters.
[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html
[16] https://www.sbert.net/

6.4 User Evaluation

To evaluate CollabCoder and answer our research questions, we conducted a within-subject user study involving 16 (8 pairs) participants who used two platforms: CollabCoder and Atlas.ti Web, for qualitative coding on two sets of qualitative data.

The goal was to address the following research questions:

- **RQ1.** Can CollabCoder support qualitative coders conduct CQA effectively?
- **RQ2.** How does CollabCoder compare to currently available tools like Atlas.ti Web?
- **RQ3.** How can the design of CollabCoder be improved?

6.4.1 Participants and Ethics

We invited 16 participants with varying qualitative analysis experiences via public channels and university email lists. We involve both experts and non-experts as lowering the bar is particularly important for newcomers or early researchers who might confront significant challenges in adhering to such rigorous workflow Richards and Hemphill, 2018; Cornish, Gillespie, and Zittoun, 2013. Among them, 2/16 participants identified as experts, 3/16 considered themselves intermediate, 4/16 as beginners, and 7/16 had no qualitative analysis experience (see details in Appendix Table B.5). Participants were randomly matched, leading to the formation of 8 pairs (see Table 6.3). Each participant received compensation of approximately $22.3 USD for their participation, based on the total duration. The study protocol was approved by our local IRB, and the financial compensation was based on the duration at the hourly rate also approved by our local IRB.

6.4.2 Datasets

We established two criteria to select the datasets used for coding task: 1) the datasets should not require domain-specific knowledge for coding, and 2) coders should be able to derive a theme tree and provide insights iteratively. Accordingly, two datasets containing book reviews on "Business" and "History" topics from the `Books_v1_00` category of `amazon_us_reviews` dataset[17] were selected. For each of them, we filtered 15 reviews to include only those with a character count between 400 and 700 and removed odd symbols such as \ and <br />. The workload was determined through pilot tests with some participants.

[17] https://huggingface.co/datasets/amazon_us_reviews/viewer/Books_v1_00/train

6.4.3 Conditions

- **Atlas.ti Web**: a powerful platform for qualitative analysis that enables users to invite other coders to collaborate by adding, editing, and deleting codes. It also allows for merging codes and generating code groups manually.
- **CollabCoder**: the formal version of our full-featured platform.

The presentation order of both platforms and materials was counter-balanced across participants using a Latin-square design (Lazar, Feng, and Hochheiser, 2017b).

6.4.4 Procedure

Each study was conducted virtually via Zoom and lasted around 2 to 3 hours. It consisted of a pre-study questionnaire, training for novice participants, two qualitative coding sessions with different conditional systems, a post-study questionnaire, and a semi-structured interview.

Introduction to the Task

After obtaining consent, we introduced the task to the pairs of participants, which involved analyzing reviews and coding them to obtain meaningful insights. We introduced research questions they should take into account when coding, such as recurring themes or topics, common positive and negative comments or opinions. We provided guidelines to ensure that the coding was consistent across all participants. Participants were permitted to use codes that were under 10 words in length, include multiple codes for each data unit, and add both descriptive and in-vivo codes.

Specific Process

Following the introduction, we provided a video tutorial on how to use the platform for qualitative coding. Participants first did independent coding, and then discussed the codes they had found and made final decisions for each unit, ultimately forming thematic groups. We urge them to engage in extensive discussions and to present code groups that accurately reflect the valuable insights they have acquired, emphasizing the importance of quality. To ensure they understand the study purpose better, participants were shown sample code groups as a reference for the type of insights they should aim to obtain from their coding. After completing the coding for all sessions, participants were asked to complete a survey, which included a 5-level Likert Scale to rate the effectiveness of two platforms, and self-reported feelings about the platforms.

Data Recording

During the process, we asked participants to share their screens and obtained their consent to record the meeting video for the entire experiment. Once the coding sessions were completed, participants were invited to participate in a post-study semi-structured interview.

6.5 Results

6.5.1 RQ1: Can CollabCoder support qualitative coders to conduct CQA effectively?

Key Findings (KF) on features that support CQA

KF1: *CollabCoder* workflow simplifies the learning curve for CQA and ensures coding independence in the initial stages. Overall, users found CollabCoder to be better as it supports side-by-side comparison of data, which makes the coding and discussion process **easier to understand (P2), more straightforward (P7), and beginner-friendly (P4)** than Atlas.ti Web, and P4 noted that CollabCoder had a **lower learning curve**.

Moreover, CollabCoder workflow preserves coding independence. Experienced users (P11 and P14), familiar with qualitative analysis, find *CollabCoder*'s independent coding feature to be particularly beneficial: *"So you don't see what the other person is coding until like both of you are done. So it doesn't like to affect your own individual coding...[For Atlas.ti Web] the fact like you can see both persons' codes and I think I'm able to edit the other person's codes as well, which I think might not be very a good practice."* Similarly, P14 indicated: *"I think CollabCoder is better if you aim for independent coding."*

KF2: Individual workspace with GPT assistance is valued for reducing cognitive burden in Phase 1. CollabCoder makes it easier for beginner users to propose and edit codes compared to Atlas.ti Web. 7/16 participants appreciated that GPT's additional assistance (P7, P15), which gave them reference (P1) and decreased thinking (P9). Such feelings are predominantly reported by individuals who are either beginners or lack prior experience in qualitative analysis. As P13 said, *"I think the CollabCoder one is definitely more intuitive in a sense, because it provides some suggestion, you might not use it, but at least some basic suggestions, whereas the Atlas.ti one, you have to take from scratch and it takes more mental load."* (P13).

Some of these beginners also showed displeasure towards GPT, largely stemming from its content summarization level, which users cannot regulate. P1 (beginner) found that **in certain instances, CollabCoder generated highly detailed summaries** which might not be well-suited to their requirements, leading them to prefer crafting their own summaries: *"One is that its summary will be very detailed, and in this case, I might not use its result, but I would try to summarize [the summary] myself."* This caused them to question AI's precision and appropriateness for high-level analysis, especially in the context of oral interviews or focus groups.

In addition, when adding codes, our participants indicated that they preferred **reading the raw data first before looking at the suggestions**, as they believed that reading the suggestions first could influence their thinking process (P1, P3, P4, P14) and introduce bias into their coding: *"So I read the text at first. it makes more sense, because like, if you were to solely base your coding on [the AI agent], sometimes its suggestions and my interpretation are different. So it might be a bit off, whereas if you were to read the text, you get the full idea as to what the review is actually talking about. The suggestion functions as a confirmation of my understanding."* (P4)

KF3: Pre-defined data units, documented decision-making mechanisms, and progress bar features collectively enhance mutual understanding in Phase 1 and Phase 2. Regarding collaboration, users found that having a pre-defined unit of analysis enabled them to more easily understand the context: *"I am able to see your quotations. Basically what they coded is just the entire unit. But you see if they were to code the reviews based on sentences, I wouldn't actually do the hard work based on which sentence he highlighted. But for CollabCoder, I am able to see at a glance, the exact quotations that they did. So it gives me a better sense of how their codes came about."* (P3) Moreover, users emphasized the importance of not only having the quotation but also keeping its context using pre-defined data units, as they often preferred to refer back to the original text. This is because understanding the context is crucial for accurate data interpretation and discussion: *"I guess, it is because like we're used to reading a full text and we know like the context rather than if we were to read like short extracts from the text. the context is not fully there from just one or two line [quotations]."* (P9)

Users also appreciated CollabCoder's keywords-support function, as it **aided them in capturing finer details (P9)** and facilitated a deeper understanding of the codes added: *"It presents a clearer view about that paragraph. And then it helps us to get a better idea of what the actual correct code should be. But since the other one [Atlas.ti Web] is [...] a little bit more like superficial, because it's based solely on two descriptive words."* (P14)

The progress bar feature in CollabCoder was seen as helpful when collaborating with others. It allowed them to **manage their time better and track the progress of each coder.** *"I actually like the progress bar because like that I know where my collaborators are."* (P8) Additionally, it acted as a **tracker to notify the user if they missed out on a part**, which can help to avoid errors and improve the quality of coding. *"So if say, for example, I missed out one of the codes then or say his percentage is at 95% or something like that, then we will know that we missed out some parts"* (P3)

All the above features collectively improve the mutual understanding between coders, which can decrease the effort devoted to revisiting the original data and recalling their decision-making processes, and deepen discussions in a limited time.

KF4: The shared workspace with metrics allows coders to understand disagreements and initiate discussions better in Phase 2. In terms of statistics during the collaboration, the similarity calculation and ranking features enable users to **quickly identify (dis)agreements (P2, P3, P7, P10, P14) to ensure they focus more (P4).** As P14 said, *"I think it's definitely a good thing [to calculate similarity]. From there, I think we can decide whether it's really a disagreement on whether it's actually two different information captured in the two different codes."* Moreover, the identification of disagreements is reported to **pave the way for discussion (P1, P8)**: *"So I think in that sense, it just opens up the door for the discussion compared to Atlas.ti...[and]better in idea generation stands and opening up the door for discussion."* (P8) In contrast, Atlas.ti necessitated more discussion initiation on the part of users.

Nevertheless, ranking similarity using CollabCoder might have a negative effect, as it may make coders focus more on improving their agreements instead of providing a more comprehensive data interpretation: *"I think pros and cons. because you will feel like there's a need to get high similarity on every code, but it might just be different codes. So there might be a misinterpretation."* (P7)

The participants had mixed opinions regarding the usefulness of IRR in the coding process. P9 found Cohen's kappa useful for their report as they do not need to calculate manually: *"I think it's good to have Cohen's Kappa, because we don't have to manually calculate it, and it is very important for our report. "* However, P6 did not consider the statistics to be crucial in their personal research as they usually do coding for interview transcripts. *"Honestly, it doesn't really matter to me because in my own personal research, we don't really calculate. Even if we have disagreements, we just solve it out. So I can't comment on whether the statistics are relevant, right from my own personal experience."* (P6)

KF5: The GPT-generated primary code groups in Phase 3 enable coders to have a reference instead of starting from scratch, thereby reducing cognitive burden. Participants expressed a preference for the automatic grouping function of CollabCoder, as it was **more efficient (P1, P2, P8, P14) and less labor-intensive (P3)**, compared to the more manual approach in Atlas.ti Web. In particular, P14 characterized the primary distinction between the two platforms as Atlas.ti Web adopts a "bottom-up approach" while **CollabCoder employs a "top-down approach".** In this context, the "top-down approach" refers to the development of "overall categories/code groups" derived from the coding decisions made in Phase 2, facilitated by GPT. This approach allows users to modify and refine elements within an established primary structure or framework, thereby eliminating the need to start from scratch. Conversely, the "bottom-up approach" means generating code groups from an existing list, through a process of reviewing, merging, and grouping codes with similar meanings. This difference impacts the mental effort required to create categories and organize codes. *"I think it's different also because Atlas.ti is more like a bottom-top approach. So we need to see through the primary codes to create the larger categories which might be a bit more tedious, because usually, they are the primary codes. So it's very hard to see an overview of everything at once. So it takes a lot of mental effort, but for CollabCoder, it is like a top-down approach. So they [AI] create the overall categories. And then from there, you can edit and then like shift things around which helps a lot. So I also prefer CollabCoder."* (P14) P1 also highlighted that this is particularly helpful when dealing with large amounts of codes, as manually grouping them one-by-one becomes nearly unfeasible.

Key Findings (KF) on collaboration behaviors with CollabCoder supports

KF6: An analysis of three intriguing group dynamics manifested in two conditions In addition to the key findings on feature utilization, we observed three intriguing collaboration group dynamics, including *"follower-leader"* (P1×P2, P5×P6), *"amicable cooperation"* (P3×P4, P7×P8, P9×P10, P13×P14, P15×P16) and *"swift but less cautious"* (P11×P12). The original observation notes are listed in Appendix Table B.6 and B.7.

The *"follower-leader"* pattern typically occurred when one coder was a novice, while the other had more expertise. Often, the inexperienced coder contributed fewer ideas or only offered support during the coding process: when using Atlas.ti Web, those "lead" coders tended to take on more coding tasks than the others since their coding tasks could not be precisely quantified. Even though both of them were told to code all the data, it would end up in a situation where one coder primarily handled the work while the other merely followed with minimal input. This pattern could also appear if the coders worked at different paces (P1×P2). As a result, the more efficient coders expressed more ideas. In contrast, CollabCoder ensures equitable participation by assigning the same coding workload to each participant and offering detailed documentation of the decision-making process via its independent coding interface. This approach guarantees that coders, even if they seldom voice their opinions directly, can still use the explicit documented information to communicate their ideas indirectly and be assessed in tandem with their collaborators. Furthermore, the suggestions generated by GPT are derived from both codes and raw data, producing a similar effect.

For *"amicable cooperation"*, the coders respected each other's opinions while employing CollabCoder as a collaborative tool to finalize their coding decisions. When they make a decision, they firstly identify the common keywords between their codes, and then check the suggestions with similar keywords to decide whether to use suggestions or propose their own final code decision. Often, they took turns to apply the final code.

For example, for the first data unit, one coder might say, *"hey, mine seems better, let's use mine as the final decision,"* and for the second one, the coder might say, *"hey, I like yours, we should choose yours [as the final decision]"* (P3×P4). In some cases, such as P13×P14, both coders generally reach a consensus, displaying no strong dominance and showing respect for each other's opinions, sometimes it is challenging to finalize a terminology for the code decision. Under this kind of condition, the coders used an LLMs agent as a mediator to find a more suitable expression that takes into account both viewpoints. Although most groups maintained similar "amicable cooperation" dynamics in Atlas.ti Web sessions, some found it challenging to adhere to their established patterns. This difficulty is attributed to the fact that such patterns are more resource-intensive. Take the P7×P8 scenario as an example: in this case, the participants encountered time management challenges, as each coding session was initially scheduled to conclude within half an hour. Participants were afforded some flexibility, allowing sessions to extend slightly beyond the initially planned duration to ensure the completion of their tasks. In the CollabCoder condition, they engaged in extensive and respectful discussions, which consequently reduced the time available for the Atlas.ti Web session. Consequently, they had to expedite the process in Atlas.ti Web. This rush resulted in a situation where only one coder assumed the responsibility of merging the codes and rapidly grouping them into thematic clusters. For this coder, to access deeper insights behind these codes, additional operations like asking why another coder has this code, and clicking more to understand which sentence it means were often not feasible within the time constraints. This absence of operations forced coders to merge data relying solely on codes, without the advantage of additional contextual insights. Consequently, this approach often leads to a "follower-leader" or "leader-takes-all" dynamic. While this simplifies the process for participants, it potentially compromises the quality of the discussion. This is also evidenced by our quantitative data in Table 6.3.

The *"swift but less cautious"* collaboration was a less desirable pattern we noticed: For P11×P12, during the merging process, they would heavily rely on GPT-generated decisions in order to finish the task quickly. This scenario highlights the concerns regarding excessive reliance on GPT and insufficient deep thinking, which can negatively impact the final quality even when GPT is used as a mediator after the codes have been produced, as defined as our initial objective. Under this pattern, the pair sadly used GPT for "another round of coding" rather than as a neutral third-party decision advice provider. In the case of this particular pair working with Atlas.ti Web, a distinct pattern emerged: P11 exhibited a notably faster pace, while P12 worked more slowly. As a result, the collaboration between the participants evolved into a "follower-leader" dynamic. In this structure, the quicker participant, P11, appeared to steer the overall process, occasionally soliciting inputs from P12.

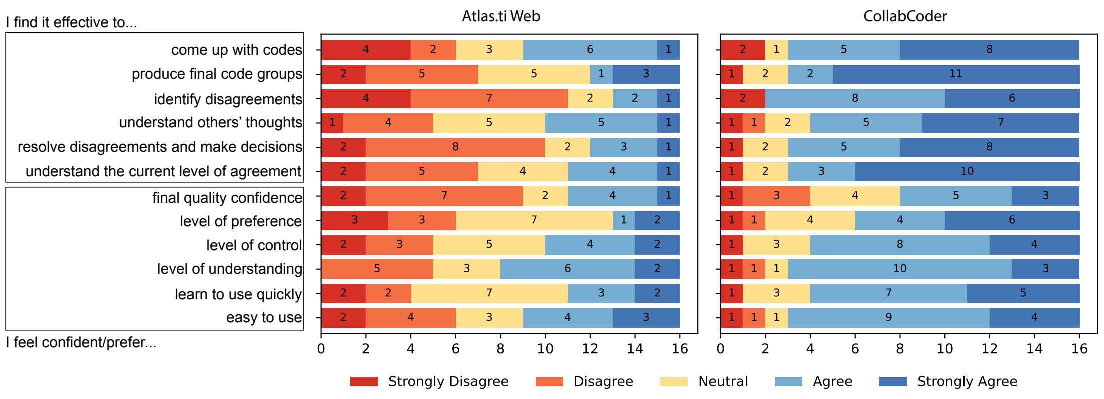


FIGURE 6.8: Post-study Questionnaires Responses from Our Participants on Different Dimensions on A 5-point Likert Scale, where 1 denotes "Strongly Disagree", 5 denotes "Strongly Agree". The numerical values displayed on the stacked bar chart represent the count of participants who assigned each respective score.

6.5.2 RQ2. How does CollabCoder compare to currently available tools like Atlas.ti Web?

Post-study questionnaire

We gathered the subjective preferences from our participants. To do so, we gave them 12 statements like *"I find it effective to..."* and *"I feel confident/prefer..."* pertaining to the effectiveness and self-perception. We then asked them to rate their agreement with each sentence on a 5-point Likert scale for each platform. The details of the 12 statements are shown in Figure 6.8.

Overall, pairwise t-tests showed that participants rated CollabCoder significantly (all $p < .05$) better than Atlas.ti Web for effectiveness in 1) *coming up with codes*, 2) *producing final code groups*, 3) *identifying disagreements*, 4) *resolving disagreements and making decisions*, 5) *understanding the current level of agreement*, and 6) *understanding others' thoughts*. The results also indicated that participants believed CollabCoder ($M = 4$) could be learned for use quickly compared to Atlas.ti Web ($M = 3.1, t(15) = -3.05, p < .01$). For other dimensions, the *confidence in the final quality*, *perceived level of preference*, *level of control*, *level of understanding*, and *ease of use*, while our results show a general trend where CollabCoder achieves higher scores, we found no significant differences. Additionally, we observed that one expert user (P6) exhibited a highly negative attitude towards implementing AI in qualitative coding, as he selected "strongly disagree" for nearly all the assessment criteria. We will discuss his qualitative feedback in Section 6.6.2.

Log data analysis

A two-tailed pairwise t-test on *Discussion Time* revealed a significant difference ($t(15) = -3.22, p = .017$) between CollabCoder ($M \approx 24mins, SD \approx 7mins$) and Atlas.ti Web ($M \approx 11mins, SD \approx 5.5mins$). **Discussions under the CollabCoder condition were significantly longer than those in the Atlas.ti Web condition.** When examining the IRR, it was found that the IRRs in the Atlas.ti Web condition were overall significantly ($t(7) = -6.69, p < .001$) lower ($M = 0.06, SD = 0.40$), compared to the CollabCoder condition ($M \approx 1$). In the latter, participants thoroughly examined all codes, resolved conflicts, merged similar codes, and reached a final decision for each data unit. Conversely, Atlas.ti Web posed challenges in comparing individual data units side-by-side, leading to minimal code discussions overall (averaging 4.5 codes discussed) compared to the CollabCoder option (averaging 15 codes discussed). Consequently, we surmise that concealed disagreements within Atlas.ti Web might require additional discussion rounds to attain a higher agreement level. Further evidence is needed to validate this assumption.

6.5.3 RQ3. How can the design of CollabCoder be improved?

While CollabCoder effectively facilitates collaboration in various aspects, as discussed in Section 6.5.1, we observed divergent attitudes toward certain functions, such as labeling certainty, relevant code suggestions, and the use of individual codebooks.

Most participants expressed concerns about the clarity, usefulness, and importance of the certainty function in CollabCoder. The self-reported nature, the potential of inconsistencies in reporting, and minimal usage among users suggest that the **certainty function may not be as helpful as intended**. For example, P12 found the certainty function "not really helpful", and P13 admitted forgetting about it due to the numerous other subtasks in the coding process. P3 also reported limited usage of the function, mainly assigning low certainty scores when not understanding the raw data. However, P14 recognized that the certainty function could be helpful in larger teams, as it might help flag quotes that require more discussion.

The perceived usefulness of the relevant code function in CollabCoder depends on the dataset and users' preferences. Some participants found it **less relevant than the AI agent's summary function**, which they considered more accurate and relevant. *"Maybe not that useful, but I think it depends on your dataset. Say whether they are many similar data points or whether they are different data points. So I think in terms of these cases they are all very different, have a lot of different contents. So it's not very relevant, but definitely, I think, in datasets which might be more relevant, could be useful."* (P2)

As for the individual codebook function, although users acknowledged its potential usefulness in tracking progress and handling large datasets, most users *"did not pay much attention to it during this coding process"* (P2, P3, P4). P3 found it helpful for tracking progress but did not pay attention to it during this particular process. P4 acknowledged that the function could be useful in the long run, particularly when dealing with a large amount of data.

While these features may not be as useful as initially anticipated, evidenced by low usage frequency or varying effectiveness across different datasets, further investigation is necessary to ascertain if the needs and challenges associated with these features truly exist or are merely perceived by us. This could significantly enhance user experiences with CollabCoder and inform the future design of AI-assisted CQA tools.

6.6 Discussion and Design Implications

We discuss users' feedback on various features, as well as the potential role that LLMs could play in the CQA workflow.

6.6.1 Facilitating Rigorous, Lower-barrier CQA Process through Workflow Design Aligned with Theories

Practically, CollabCoder contributes by providing a one-stop, end-to-end workflow that ensures seamless data transitions between stages with minimal effort. This design is grounded in qualitative analysis theories such as Grounded Theory (Flick, 2013) and Thematic Analysis (Maguire and Delahunt, 2017), as outlined in Section **??**, facilitating a rigorous yet accessible approach to CQA practice. While spreadsheets are also capable of similar processes, they typically demand considerable effort and struggle to uphold a stringent process due to the intricacy and nuances involved. CollabCoder, in contrast, streamlines these tasks, rendering the team coordination process (Entin, 2000; Malone and Crowston, 1994) more practical and manageable. Our evaluation demonstrates the effectiveness of CollabCoder, empowering both experienced practitioners and novices to perform rigorous and comprehensive qualitative analysis.

Apart from practical benefits, our CollabCoder design (Cash, 2018) can also enrich theoretical understanding in the CQA domain (Jörg Hecker, 2023), which aids practitioners in grasping foundational theories, thereby bolstering the credibility of qualitative research (Collins and Stockton, 2018; Jörg Hecker, 2023). Over the years, CQA practices have remained inconsistent and vague, particularly regarding when and how multiple coders may be involved, the computation of IRR, the use of individual coding phases, and adherence to existing processes (Bradley, 1993; Noble and J. Smith, 2015). A common question could arise: *"If deviating from strict processes does not significantly impact results, or the influence is hard to perceive (at least from others' perspective), why should substantial time be invested in maintaining them, especially under time constraints?"* Current software like Atlas.ti, MaxQDA often neglects this critical aspect in their system design, focusing instead on basic functionalities like data maintenance and code addition, which, are not the most challenging parts of the process for practitioners. Ultimately, CollabCoder enables practitioners to conduct a CQA process that is both transparent and standardized within the community (Noble and J. Smith, 2015; Moravcsik, 2014). Looking forward, we foresee a future where coders, in documenting their methodologies, will readily reference their use of such specifically designed workflows or systems for CQA analysis.

With this in mind, our objective is not to position any single method as the definitive standard in this field. Although CollabCoder is specifically designed for one type of coding — consensus coding within inductive coding — we do not exclusively advocate for either consensus or split coding. Instead, we emphasize that coders should choose a method that aligns best with their data and requirements (Teherani et al., 2015; Jörg Hecker, 2023; Grad Coach, 2023; Collins and Stockton, 2018). Therefore, the design of such tools should aim to accommodate various types of qualitative analysis methods. For instance, split coding might necessitate distributing data among team members in a manner that differs from the uniform distribution required by consensus coding.

6.6.2 LLMs as “Suggestion Provider” in Open Coding: Helper, not Replacement.

Utilizing LLMs to Reduce Cognitive Burden

Independent open coding is a highly cognitively demanding task, as it requires understanding the text, identifying the main idea, creating a summary based on research questions, and formulating a suitable phrase to convey the summary (Lazar, Feng, and Hochheiser, 2017b; J. Corbin and Strauss, 2008). Additionally, there is the need to refer to and reuse previously created codes. In this context, GPT’s text comprehension and generation capabilities can assist in this mentally challenging process by serving as a suggestion provider.

Improving LLMs’ Suggestions Quality

However, a key consideration according to KF2 is how GPT can provide better quality suggestions that align with the needs of users. For CollabCoder, we only provided essential prompts such as "summary" and "relevant codes". However, a crucial aspect of qualitative coding is that coders should always consider their research questions while coding and work towards a specific direction. For instance, are they analyzing the main sentiment of the raw data or the primary opinion regarding something? This factor can significantly impact the coding approach (e.g., descriptive or in-vivo coding (Saldaña, 2021)) and what should be coded (e.g., sentiment or opinions). Therefore, the system should support mechanisms for users to inform GPT of the user’s intent or direction. One possible solution is to include the research question or intended direction in the prompt sent to GPT alongside the data to be coded. Alternatively, users could configure a customized prompt for guidance, directing GPT's behavior through the interface (Ippolito et al., 2022). This adaptability accommodates individual preferences and improves the overall user experience.

Looking ahead, as the underlying LLM evolves, we envision that an approach for future LLM assistance in CollabCoder involves: 1) creating a comprehensive library of both pre-set and real-time updated prompts, designed to assist in suggesting codes across diverse fields like psychology and HCI; 2) implementing a feature that allows coders to input custom prompts when the default prompts are not suitable.

LLMs should Remain a Helper

Another key consideration is how GPT can stay a reliable suggestion provider without taking over from the coder (J. A. Jiang et al., 2021; Marathe and Toyama, 2018a). Our study demonstrated that both novices and experts valued GPT's assistance, as participants used GPT's suggestions either as code or as a basis to create codes 76.67% of the time on average.

However, one expert user (P6) held a negative attitude towards employing LLMs in open coding, assigning the lowest score to nearly all measures (see Figure 6.8). This user expressed concerns about the role of AI in this context, suggesting that qualitative researchers might feel forced to use AI-generated codes, which could introduce potential biases. Picking up the nuances from the text is considered *"fun"* for qualitative researchers (P6), and suggestions should not give the impression that *"the code is done for them and they just have to apply it"* (P6) or lead them to *"doubt their own ideas"* (P5).

On the other side, it is important not to overlook the risk of over-reliance on GPT. While we want GPT to assist, we do not intend for it to fully replace humans in the process, as noted in DG5. Our observations revealed that although participants claimed they would read the raw data first and then check GPT's suggestions, some beginners tended to rely on GPT for forming their suggestions, and experts would unconsciously accept GPT's suggestions if unsure about the meaning of the raw data, in order to save time. Therefore, preserving the enjoyment of qualitative research and designing for appropriate reliance (J. D. Lee and See, 2004) to avoid misuse (Dzindolet et al., 2003) or over-trust can be a complex challenge (Xiao et al., 2023). To this end, mixed-initiative systems (J. Allen, Guinn, and Horvtz, 1999; Horvitz, 1999) like CollabCoder can be designed to allow for different levels of automation. For example, GPT-generated suggestions could be provided only for especially difficult cases upon request, rather than being easily accessible for every unit, even when including a pre-defined time delay.

6.6.3 LLMs as "Mediator" and "Facilitator" in Coding Discussion

Among the three critical CQA phases we pinpointed, aside from the open coding phase, the subsequent two stages — Phase 2 (merge and discussion) and Phase 3 (development of a codebook) — require a shared workspace for coders to converse. We took note of the role LLMs undertook during these discussions.

LLMs as a "Mediator" in Group Decision-Making.

The challenge of dynamically reaching consensus — a decision that encapsulates the perspectives of all group members — has garnered attention in the HCI field (Emamgholizadeh, 2022; Pérez et al., 2018; J. A. Jiang et al., 2021). Jiang et al. (J. A. Jiang et al., 2021) extensively explore collaborative dynamics in their research for qualitative analysis. They highlighted decision-making modes in consensus-building may vary under different power dynamics (Interaction Institute for Social Change, 2018) in CQA context. In some cases, the primary author or a senior member of a project may assume the decision-making role. According to our KF6, we also found interesting group dynamics, identifying patterns like "amicable cooperation", "follower-leader", and "swift but less cautious" modes. Our design positions GPT as a mediator or a group recommendation system (Emamgholizadeh, 2022), particularly useful when consensus is hard to reach. In this role, GPT acts as an impartial facilitator, aiding in harmonizing labor distribution and opinion expression. It guides groups towards decisions that are not only cost-effective but also equitable, justified, and sound (L. Chen et al., 2013). This is a functionality that can hardly be achieved by using tools like Atlas.ti Web. In fact, these group dynamics can also be explored through various lenses, such as the Thomas-Kilmann conflict modes (Thomas, 2008), which emphasize the importance of balancing assertiveness and cooperativeness in a team. Delving into these theories can significantly aid in the design of more effective team collaboration tools.

Nonetheless, CollabCoder's present design in Phase 2, which employs LLMs as a recommendation system for coding decisions, represents merely an initial step. While the CollabCoder cannot fundamentally alter collaborative power dynamics, it ensures that coding is a collaborative effort, emphasizing substantive discussions between two coders to avoid superficial collaboration. Looking ahead, there are numerous paths we could and should pursue. For example, as humans should be the ultimate decision-makers, with GPT serving merely as a fair mediator between coders, group decision recommendations ought to be made available only upon explicit request. Alternatively, once a coder puts forth a final decision, GPT could then refine the wording or formulate some conclusive description to facilitate future reflection on the code decisions (Barry et al., 1999).

LLMs as "Facilitator" in Streamlining Primary Code Grouping

As per KF5, our participants offered insightful feedback about using GPT to generate primary code groups. They found the top-down approach, where GPT first generates primary groups and users subsequently refine and revise them, more efficient and less cognitively demanding compared to the traditional bottom-up method. In the traditional method, users must begin by examining all primary codes, merging them, and then manually grouping them into categories, which can be mentally taxing. Differently, CollabCoder is designed to initially formulate primary or coarse ideas about how to group codes. Similar to many types of recommendation systems, the suggestions provided by CollabCoder are intended to complement the coders' initial thoughts on code grouping. When coders review these GPT-suggested code groups, it enables them to reflect upon and compare their own ideas with the given suggestions. This process enriches the final code groups by efficiently incorporating a wider range of perspectives, extending beyond the insights of just the two coders. This ensures a more comprehensive and multifaceted categorization. Moreover, researchers can more effectively and easily manage large volumes of data and potentially enhance the quality of their analysis.

However, it is crucial to exercise caution when applying this method. We observed that when time constraints exist, coders may skip discussions, with only one of two coders combining and categorizing the codes into code groups (P7×P8). Additionally, P14 mentioned that GPT appears to dominate the code grouping process, resulting in a single approach to grouping. For instance, while the participants might create code groups based on sentiment analysis during their own coding process, they could be tempted to focus on content analysis under GPT's guidance.

Similarly, to overcome these challenges of CollabCoder, we envision a system where coders would create their own groupings first and only request LLMs' suggestions afterward. Alternatively, LLMs' assistance could be limited to situations where the data volume is substantial. Another approach could be prompting LLMs to generate code groups based on the research questions rather than solely on the (superficial) codes. This would ensure a more contextually relevant and research-driven code grouping process.

6.7 Limitations and Future Work

This work has limitations. **Firstly, it's important to note that the current version of CollabCoder operates under certain assumptions, deeming coding tasks as "ideal"—comprising semantically independent units, a two-person coding team, and data units with singular semantics.** However, our expert interviews revealed a more complex reality. One primary source of disagreement arises when different users assign multiple codes to the same data unit, often sparking discussions during collaborative coding. Future research should aim to address this point.

Secondly, we only used pre-defined unit data and did not consider splitting complex data into units (e.g., interview data). Future work could explore utilizing GPT to support the segmentation of interview data into semantic units and automating the import process.

Lastly, we did not investigate the specific process by which users select and edit a GPT suggestion. Future research could delve deeper into how users incorporate these suggestions to generate a final idea. Moreover, for a tool that could be used by the same coder on multiple large datasets, it would also be beneficial to have GPT generate suggestions based on users' coding patterns rather than directly providing suggestions.

6.8 Conclusion

This paper introduces CollabCoder, a system that integrates the key stages of the CQA process into a one-stop workflow, aiming to lower the bar for adhering to a strict CQA procedure. Our evaluation with 16 participants indicated a preference for CollabCoder over existing platforms like Atlas.ti Web due to its user-friendly design and AI assistance tailored for collaboration. We also demonstrated the system's capability to streamline facilitate discussions and consensus-building, and create codebooks. By examining both human-AI and human-human interactions within the context of qualitative analysis, we have uncovered key challenges and insights that can guide future design and research.

TABLE 6.3: Overview of the final coding results. "Collab." denotes CollabCoder, "Atlas." denotes Atlas.ti Web, "Total Codes" denotes the total number of codes generated while "Discussed Codes" denotes the total number of codes that were discussed by the coders during the discussion phase. "Bus." denotes the "Business" dataset while "His." denotes the "History" dataset. "Suggestions Acceptance" column denotes the proportion of usage of GPT-generated codes (GPT), the selection from the relevant codes in code history suggested by GPT (Rele.), and users' self-proposed codes (Self.) to the total number of open codes in Phase 1. "GPT-based Code Decisions" column reflects the proportion of code decisions in Phase 2 that originated from suggestions made by the GPT mediator.

Pairs	Self-reported QA expertise	Conditions	Collaboration Observation	Total Codes		Discussed Codes (No.)		IRR (-1 to 1)		Code Groups (No.)		Discussion Time (mins:secs)		Suggestions Acceptance in Phase 1 (%)			GPT-Based Code Decisions (%)
				Collab.	Atlas.	Collab.	Atlas.	Collab.[b]	Atlas.	Collab.	Atlas.	Collab.	Atlas.	GPT	Rele.	Self.	
P1	Beginner	Atlas. (Bus.),	Follower-	15	24	15	6	NA	-0.07	6	3	19:41	07:39	100	0	0	5
P2	No Experience	Collab.His.)	Leader											70	5	25	
P3	Expert	Collab.(Bus.),	Amicable	15	10	15	10	NA	1	5	4	35:24	21:32	90	0	10	40
P4	No Experience	Atlas. (His.)	Cooperation											90	0	10	
P5	No Experience	Atlas. (His.),	Follower-	15	11	15	2	NA	-0.02	5	2	17:55	06:16	73	7	20	100
P6	Expert	Collab.(Bus.)	Leader											100	0	0	
P7	No Experience	Collab.(His.)	Amicable	15	22	15	2	NA	-0.33	7	6	29:08	No discussion[a]	7	0	93	80
P8	No Experience	Atlas. (Bus.)	Cooperation											13	7	80	
P9	Intermediate	Atlas. (Bus.),	Amicable	15	17	15	5	NA	0.04	5	2	15:11	14:38	73	13	13	80
P10	No Experience	Collab.(His.)	Cooperation											53	40	7	
P11	Intermediate	Collab.(Bus.),	Quick and	15	61	15	2	NA	-0.07	3	3	19:23	14:15	100	0	0	100
P12	No experience	Atlas. (His.)	not careful											100	0	0	
P13	Beginner	Atlas. (His.),	Amicable	15	30	15	5	NA	-0.08	8	2	29:19	08:43	87	7	7	100
P14	Intermediate	Collab.(Bus.)	Cooperation											93	0	7	
P15	Beginner	Collab.(His.)	Amicable	15	8	15	4	NA	0.04	4	2	29:09	08:52	100	0	0	43
P16	No experience	Atlas. (Bus.)	Cooperation											73	20	7	
Mean				15	22.88	15	4.5	NA	0.06	5.38	3	24:00	10:48	76.46	6.15	17.4	68.5
SD				0	17.19	0	2.73	NA	0.40	1.60	1.41	07:12	05:24	29.43	10.74	28.11	35.3

[a] P7 and P8 gave up discussion for the Atlas.ti session due to spending too much time in the CollabCoder session.
[b] Following the discussion session in CollabCoder, the original codes have been restructured and finalized as a single code decision, resulting in an IRR≈1. Consequently, IRR calculations are not applicable (NA) for the CollabCoder conditions.

Chapter 7

Discussion and Future Work

In the future, I intend to extend my research on leveraging HCI methods to enhance human-AI collaboration and other areas like software engineering. Below, I highlight several research themes that particularly excite me.

7.1 Towards Automating Qualitative Analysis with Large Language Models

In this thesis, we have focused exclusively on systems that integrate AI to assist humans in qualitative analysis. However, a pertinent question arises: Is it feasible for LLMs to autonomously perform qualitative analysis if provided with specific research questions? The answer remains uncertain. Previous research has made progress in automatically prompting GPT for data-related tasks, particularly in the domain of HCI. This includes studies where GPT models have been utilized for synthesizing HCI data (Hämäläinen, Tavast, and Kunnari, 2023), applying these models to deductive coding with predetermined codebooks (Xiao et al., 2023), and examining their potential to function as independent researchers in data interpretation (Byun, Vasicek, and Seppi, 2023).

Numerous unanswered questions remain regarding GPT's capabilities in qualitative analysis. For instance, it's unclear whether GPT can autonomously conduct inductive qualitative analysis, such as developing a coding schema from unorganized data. Further research questions we could explore include:

- To what degree do GPT-generated schemas align with those created by human analysts?
- Can GPT-generated coding schemas be effectively integrated into subsequent qualitative coding processes?
- What strategies can be employed to prompt GPT to iteratively refine a coding schema, making it suitable for deductive coding?
- Before interpretation, how might GPT assist in data cleaning to streamline subsequent data analysis?

- Can we leverage the depth and systematic nature of qualitative analysis to mitigate the challenges, particularly the ambiguity problem, inherent in data annotation tasks in Natural Language Processing (NLP)?

These questions highlight the need for in-depth exploration into the practical applications and limitations of GPT in qualitative research.

7.2 Constructing Frameworks of Human-LLM Collaboration

Large language models (LLMs) like ChatGPT enable users to engage in a dialogue, allowing for follow-up questions, corrections of its errors, challenging inaccurate premises, and denying inappropriate requests. Since ChatGPT's introduction, conversational interaction has become the predominant mode between humans and LLMs. This shift necessitates that humans craft diverse, effective prompts to accomplish tasks. Despite well-designed prompts enhancing LLMs' capabilities in many areas, I have noticed that the mere conversational interaction can sometimes fail in addressing intricate tasks, encompassing reasoning, organization, and creativity.

As such, optimizing human-LLM interaction may be pivotal in elevating these abilities further. While many frameworks have been proposed to guide the analysis, design, and critique of human-AI applications in a theoretical context, LLMs differ from traditional AI in terms of human-AI interaction modes. Specifically, when considering interactions between humans and LLMs, they lean heavily on manipulating complex and abstract levels of prompts. Therefore, I believe this emergence of new design paradigms has exposed a gap, emphasizing the necessity for a design framework tailored to this distinct context.

7.3 Augmenting LLMs for Other Areas like Code Auditing

Building on the previously mentioned framework, I found that the design models for enhancing GPT to handle intricate tasks have versatile applications, including code auditing. In the realm of code auditing, GPT has emerged as a tool for identifying security vulnerabilities, especially within smart contracts. These contracts are often susceptible to a myriad of vulnerabilities. Though many tools exist that target known patterns, they fall short as they fail to detect around 80% of Web3 security bugs.

Recognizing this gap, Sun et al. (Sun et al., 2023) introduced GPTscan. This method synergizes GPT with static analysis to detect previously elusive vulnerabilities. While GPT inherently grasps code vulnerabilities and identifies pertinent attributes, its matching can sometimes be imprecise. Consequently, GPTScan incorporates a novel filtering process, honing in on the most relevant candidate functions for GPT matching.

From this, I found that in code auditing, especially with tools like GPTscan, the success is not just about improving GPT's capabilities. A significant opportunity lies in human-AI collaboration as addressing challenges often requires human insights and feedback, for instance, in designing filters that rely on human feedback to minimize false positives. A targeted approach would be to apply HCI methods like interviews and interaction design to develop interactive tools. Therefore, a synergy between LLMs' code understanding ability strength and human intuition can lead to more accurate and comprehensive vulnerability detection.

Chapter 8

Conclusion

With the burgeoning advancement of AI and LLMs, the central question driving my PhD research is: 'How can we enhance collaboration in qualitative analysis—both in human-to-human interactions through AI and in direct human-AI partnerships?' This thesis delves into this question, structured in two distinct yet interconnected parts: 1) establishing innovative collaborative workflows tailored for qualitative coding teams that include multiple humans and AI models; 2) enhancing the trust and reliance dynamics of human-AI teams by examining their varied interactions. This is achieved through lab-based experiments, interviews, surveys, and various other methods used in HCI research (Lazar, Feng, and Hochheiser, 2017a).

In Chapter 2: This chapter explains the context surrounding the selection of my research topic is thoroughly explained. This chapter delves into the foundational motivations driving this thesis, offering insight into the reasons behind the chosen area of study at that time.

In Chapter 3: This chapter introduces the core theoretical frameworks underpinning this thesis. It also presents an overview of related work in the field of generative AI, particularly focusing on its applications and implications in qualitative analysis.

In Chapter 4: We explore AI's role in enhancing human-to-human collaboration in coding, a novel research area. This chapter details the process of understanding coder behaviors and challenges through interviews and the development of CoAIcoder, a prototype that offers coding suggestions based on historical data. We evaluate various collaborative approaches and discover key trade-offs between coding efficiency and quality, examining how different levels of independence affect outcomes in qualitative coding scenarios.

In Chapter 5: The chapter investigates how the granularity of code and text impacts user trust in AI-assisted qualitative coding (AIQC). We conduct a detailed study with a split-plot design involving 30 participants and a follow-up with 6 participants. The findings highlight the complexity of human-AI interaction in open coding, emphasizing the need for tailored design approaches for different subtasks. We also address the issues of varying trust levels, over-reliance, and under-reliance on AI in coding, laying groundwork for future research on AIQC user trust and reliance.

In Chapter 6: Here, we introduce CollabCoder, an innovative system designed to streamline the qualitative coding process. The chapter presents an evaluation of CollabCoder against traditional platforms like Atlas.ti Web, with 16 participants showing a preference for its user-friendly interface and AI-enhanced collaborative features. The system not only facilitates efficient discussions and consensus-building but also aids in creating codebooks. This chapter sheds light on the challenges and insights in human-AI and human-human interactions, guiding future design and research in the field of qualitative analysis.

Our long-term vision extends beyond merely creating AI-assisted coding systems. We aim to develop systems that are in harmony with qualitative analysis frameworks, integrating their strengths into our designs. We encourage a broad spectrum of researchers to join this burgeoning field to address the outstanding questions we have identified. Furthermore, by leveraging the benefits of qualitative analysis methods, we aspire to enhance various domains, including data annotation. Our goal is to bridge the gap between Human-Computer Interaction (HCI) and Natural Language Processing (NLP), fostering a more integrated approach to these interrelated fields.

Appendix A

Appendix for CoAIcoder

A.1 Study Protocol

A.1.1 Welcome to AIQA Study!

Qualitative analysis (QA), a common method in human-computer interaction and social computing research, involves a key process known as coding. This procedure is crucial for discerning patterns and extracting insights from qualitative data, though it's traditionally labor-intensive and time-consuming. In recent years, researchers have introduced Artificial Intelligence (AI) to enhance the efficiency of this process. However, they've largely overlooked the collaborative aspect of the coding process. Our project seeks to bridge this gap by offering an AI-based tool to streamline collaboration among coders. Utilizing AI to facilitate this interaction could potentially improve coding efficiency, potentially saving considerable time for QA researchers.

A.1.2 Task Introduction

You are a pair of researchers who are trying to perform qualitative analysis on interview transcripts of students undergoing a preparatory mock interview. The research question is to find general qualities of candidates (include good and bad ones).

Your task is to code the sentences so that we may obtain a meaningful analysis of the transcripts. Here's an example. If we were to analyse meeting transcripts, a likely thing to be said during a meeting might be:

"Ok, can Alice please follow up with Bob on the designs".

- A reasonable way to code this sentence could be **Action Items**.
- A succinct subcode or description could be **"A person was asked to follow up on a task"**.
- This sentence would then be added as one of the **examples** of Action Items code.

Participants are also presented with a sample codebook table, specifically Table 1 from DeCuir-Gunby et al.'s work (DeCuir-Gunby, Marshall, and McCulloch, 2011).

A.1.3 Introduction to Three Phases

In the introduction, the instructor presents various strategies for employing CoAIcoder, which are designed to support distinct conditions across three phases of CQA.

A.1.4 Post-Study Interview Questions

1. What challenges have you encountered during the labeling process when working individually?
2. What difficulties arise when you engage in collaborative labeling?
3. In your opinion, how effectively does CoAIcoder manage these collaborative challenges?
4. What is your reaction when you encounter a code in the code list that you did not personally contribute?
5. How would you describe your level of confidence when using the AIcoder?
6. Has CoAIcoderf assistance proven beneficial in resolving conflicts that arise during the coding process? If so, how?
7. How frequently do you utilize CoAIcoder, and what motivates this usage?
8. Do you perceive that CoAIcoder enhances your coding efficiency and collaboration? Could you please elaborate?
9. Have you noticed a speed increase in the coding process after the formation of the codebook? If so, how has this been achieved?
10. Any other relevant and improvised questions.

A.2 Intuitive comparison of the results across four conditions

To facilitate an intuitive comparison of the results across four conditions, the table below presents the ranking of codes in the formed codebook for each condition.

TABLE A.1: The Ranking of Codes in Formed Codebook of Four Conditions. Only codes in the first level were counted. The codes in every cell are different expressions of one core idea labelled in bold.

Ranking	Condition A: Without AI, Asynchronous, not Shared Model	Condition B: With AI, Asynchronous, not Shared Model	Condition C: With AI, Asynchronous, Shared Model	Condition D: With AI, Synchronous, Shared Model
1	**Leadership**: Leadership skills(3); Leadership; leadership experiences(2); Leadership experience with poor relevance; Work Experience	**Strengths**: Humble; Motivated; Open-minded; Sociable; very focused; Courage; Reflective; Introspection; Rational; Confidence; Good qualities	**Leadership**: Leadership training; Leadership skills; Leadership Qualities(2); leadership(2); Leadership role; Leadership experience(2)	**Leadership**: Leadership experience(4); leadership(3); Leadership skills
2	**Weakness**: Using their weakness to their advantages; Weakness that is addressed; Weakness(2); Discussion on weakness; Overcoming weakness; Poor example of overcoming weakness; Weakness and overcoming weakness	**Weakness**: Weakness(6); slightly impulsive; Indecisive; Shy personality; Bad qualities; overcoming; Overcoming weakness; ways to overcome weaknesses	**Weakness**: Weakness(5); Candidate's weakness; Overcoming weakness; Weakness and how you overcome	**Weakness**: Weakness(4); personal weaknesses; Weakness and overcoming weakness; How to overcome weakness; Sharing weaknesses
3	**Hiring**: Key qualities for hiring; Irrelevant reasons for hiring; Self-marketing; Why candidate should be hired; Strengths and reason for hire; Reason to hire	**Leadership**: leadership(2); Leadership training; Group-oriented leadership style; Leadership and teamwork; leadership experience(4); Leadership skills	**Introduction**: Background; Introduction(3); self-introduction(3)	**Introduction**: Introduction(3); Introduction and Interest; Current status; self-introduction
4	**Introduction**: Personal introduction; Introduction; Education; Introduction of candidate; Interests	**Introduction**: Introduction(3); Background Information	**Challenges**: Challenges faced(3); Challenging Situation; challenge; Problem recognition; Team working challenge; Lack of resources; Language barriers	**Teamwork**: Teamwork(2); Teamwork experience(2)
5	**Teamwork**: Tendency to help/accommodate teammates; Teamwork(2); Team experience	**Challenges**: Difficulties; Challenges faced; examples of challenges working in a team; Teamwork challenges	**Hiring**: Reasons to Hire Candidate; reason to hire; perfecting herself; Reasons that interviewee should get hired; Why should you be hired	**Hiring**: Hiring Quality; Reasons to hire(2); Hiring decision; Strengths and reason for hire
6	**Problem solving**: Problem solving; Problem Solving Skills with poor relevance; Experienced problem solving skills	**Hiring**: Reasons for hiring; reason to hire; reasons for hiring interviewee	**Problem solving**: Overcome challenges(2); Dealing with the challenge; Problem-solving skills; Problem solving(2)	**Strengths**: Sharing personal strengths; Positive attributes; Strengths; Competitive advantage
7	**Interest in role**: Candidate's Vague Interest in Role; Candidate has Interest in the Role	**Interest**: Keen in health; interest	**Strengths**: Strengths(2); Candidate's strengths;	**Challenges**: Challenges faced; Challenge; Challenges and actions taken; handling challenges

Appendix B

Appendix for CollabCoder

B.1 Different CQA Software

TABLE B.1: Different CQA Software. Note: This list is based on public online resources and not exhaustive.

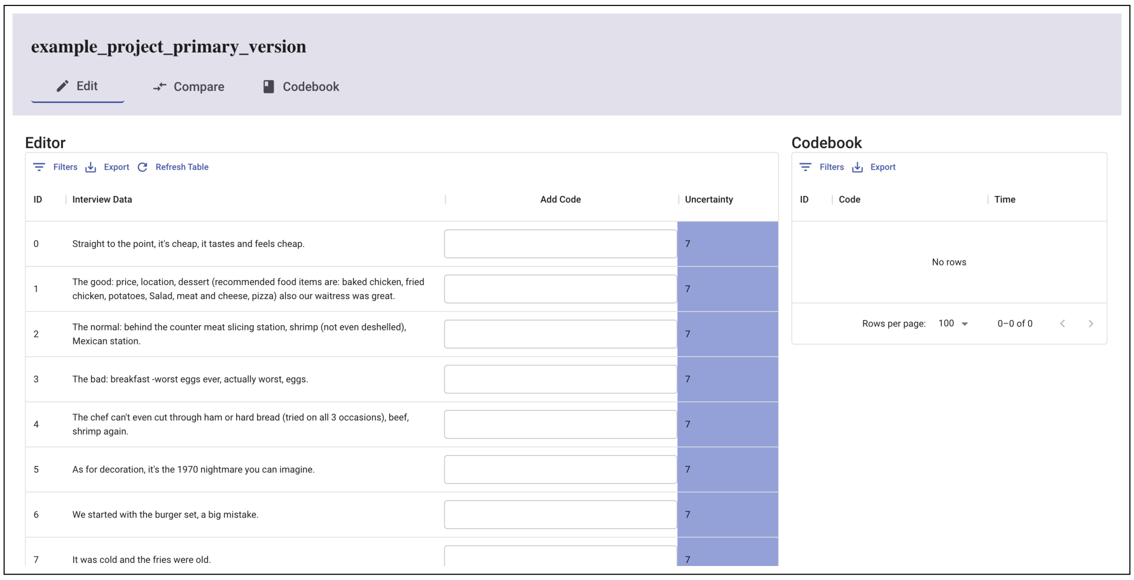
Application	Atlas.ti Desktop	Atlas.ti Web	nVivo Desktop	Google docs	MaxQDA
Collaboration ways	Coding separately and then export the project bundles to other coders	Coding on the same web page	Coding separately and then export the project bundles to other coders	Collaborative simultaneously	Provide master project that includes documents and primary codes, and then send copies to others, allowing them to merge
Coding phase	All Phases	All phases	All Phases	All phases	All Phases
Independence	independent	not independent	Inpedendent	not independent	Inpedendent
Synchrony	Asynchronous	Synchronous	Asynchronous	Synchronous	Asynchronous
Unit of analysis	Select any text	Select any text	Select any text, but calculation of IRR can be on character, sentence, paragraph	Select any text	Select any text
IRR	Agreement Percentage; Holsti Index; Krippendorff's family of Alpha	NA	Agreement Percentage; Kappa coefficient	NA	Agreement Percentage; Kappa coefficient
Calculation of IRR	Calculating after coding system is stable and all codes are defined	Calculating manually at any time	Calculating after coding system is stable and all codes are defined	NA	Calculating after coding system is stable and all codes are defined
Multi-valued coding	support multiple codes	support multiple codes	support multiple codes	support multiple codes	support multiple codes
Uncertainty/ Disagreements	NA	NA	quickly identify areas of agreement and disagreement within the source data using the green, yellow, and blue indicators on the scroll bar.	NA	NA

B.2 The primary version of CollabCoder

B.3 Prompts used in CollabCoder

B.4 Demographics of Participants

B.5 Observation notes for participants

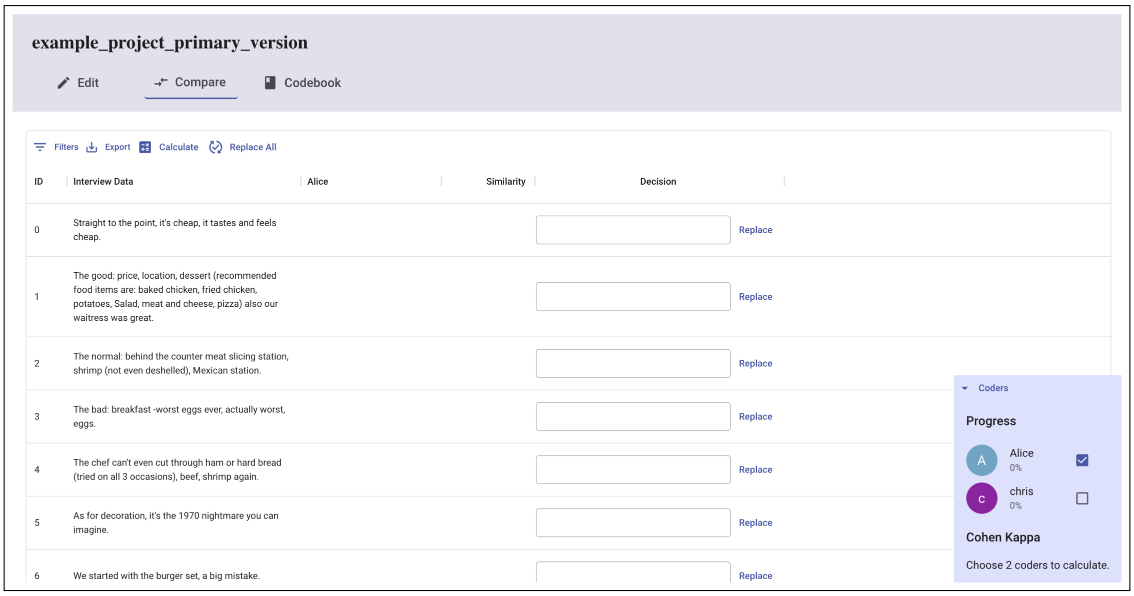


ID	Interview Data	Add Code	Uncertainty
0	Straight to the point, it's cheap, it tastes and feels cheap.		7
1	The good: price, location, dessert (recommended food items are: baked chicken, fried chicken, potatoes, Salad, meat and cheese, pizza) also our waitress was great.		7
2	The normal: behind the counter meat slicing station, shrimp (not even deshelled), Mexican station.		7
3	The bad: breakfast -worst eggs ever, actually worst, eggs.		7
4	The chef can't even cut through ham or hard bread (tried on all 3 occasions), beef, shrimp again.		7
5	As for decoration, it's the 1970 nightmare you can imagine.		7
6	We started with the burger set, a big mistake.		7
7	It was cold and the fries were old.		7

ID	Code	Time
	No rows	

Rows per page: 100 0–0 of 0

FIGURE B.1: Primary Prototype for Phase 1.



ID	Interview Data	Alice	Similarity	Decision	
0	Straight to the point, it's cheap, it tastes and feels cheap.				Replace
1	The good: price, location, dessert (recommended food items are: baked chicken, fried chicken, potatoes, Salad, meat and cheese, pizza) also our waitress was great.				Replace
2	The normal: behind the counter meat slicing station, shrimp (not even deshelled), Mexican station.				Replace
3	The bad: breakfast -worst eggs ever, actually worst, eggs.				Replace
4	The chef can't even cut through ham or hard bread (tried on all 3 occasions), beef, shrimp again.				Replace
5	As for decoration, it's the 1970 nightmare you can imagine.				Replace
6	We started with the burger set, a big mistake.				Replace

Coders

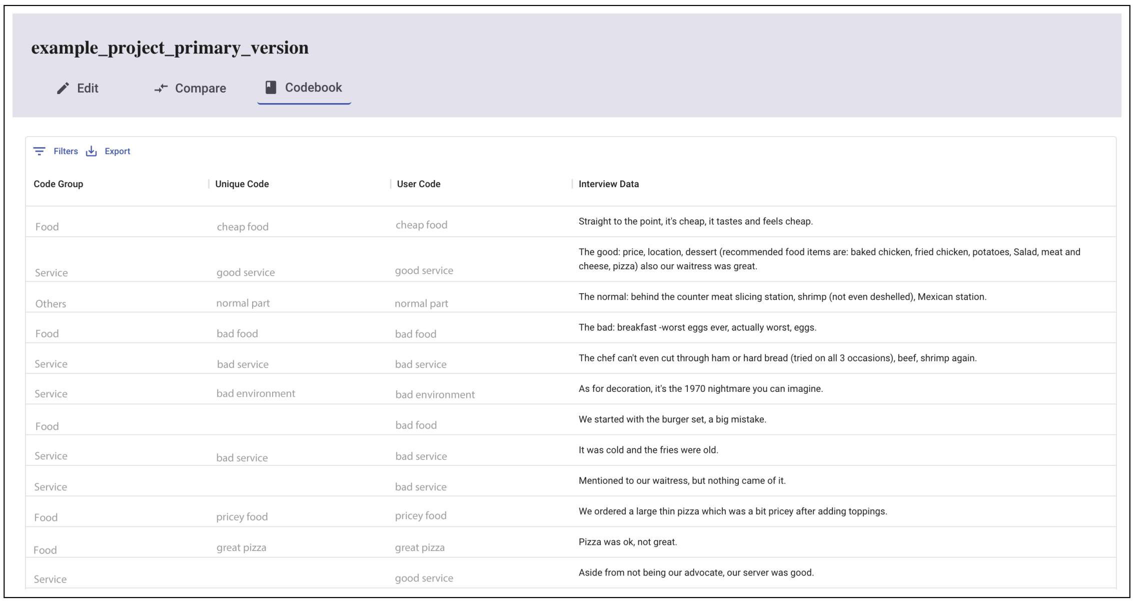
Progress

A	Alice 0%	☑
c	chris 0%	☐

Cohen Kappa

Choose 2 coders to calculate.

FIGURE B.2: Primary Prototype for Phase 2.



Code Group	Unique Code	User Code	Interview Data
Food	cheap food	cheap food	Straight to the point, it's cheap, it tastes and feels cheap.
Service	good service	good service	The good: price, location, dessert (recommended food items are: baked chicken, fried chicken, potatoes, Salad, meat and cheese, pizza) also our waitress was great.
Others	normal part	normal part	The normal: behind the counter meat slicing station, shrimp (not even deshelled), Mexican station.
Food	bad food	bad food	The bad: breakfast -worst eggs ever, actually worst, eggs.
Service	bad service	bad service	The chef can't even cut through ham or hard bread (tried on all 3 occasions), beef, shrimp again.
Service	bad environment	bad environment	As for decoration, it's the 1970 nightmare you can imagine.
Food		bad food	We started with the burger set, a big mistake.
Service	bad service	bad service	It was cold and the fries were old.
Service		bad service	Mentioned to our waitress, but nothing came of it.
Food	pricey food	pricey food	We ordered a large thin pizza which was a bit pricey after adding toppings.
Food	great pizza	great pizza	Pizza was ok, not great.
Service		good service	Aside from not being our advocate, our server was good.

FIGURE B.3: Primary Prototype for Phase 3. The gray-colored codes serve as an example to illustrate the differences between "Code Group", "Unique Code", and "User Code". The interfaces shown above, being preliminary mockups, were utilized to gather feedback from our primary interviewees in Step3, Section 6.2.1, for the refinement of the final version of interfaces including Figure 6.5, 6.6, and 6.7.

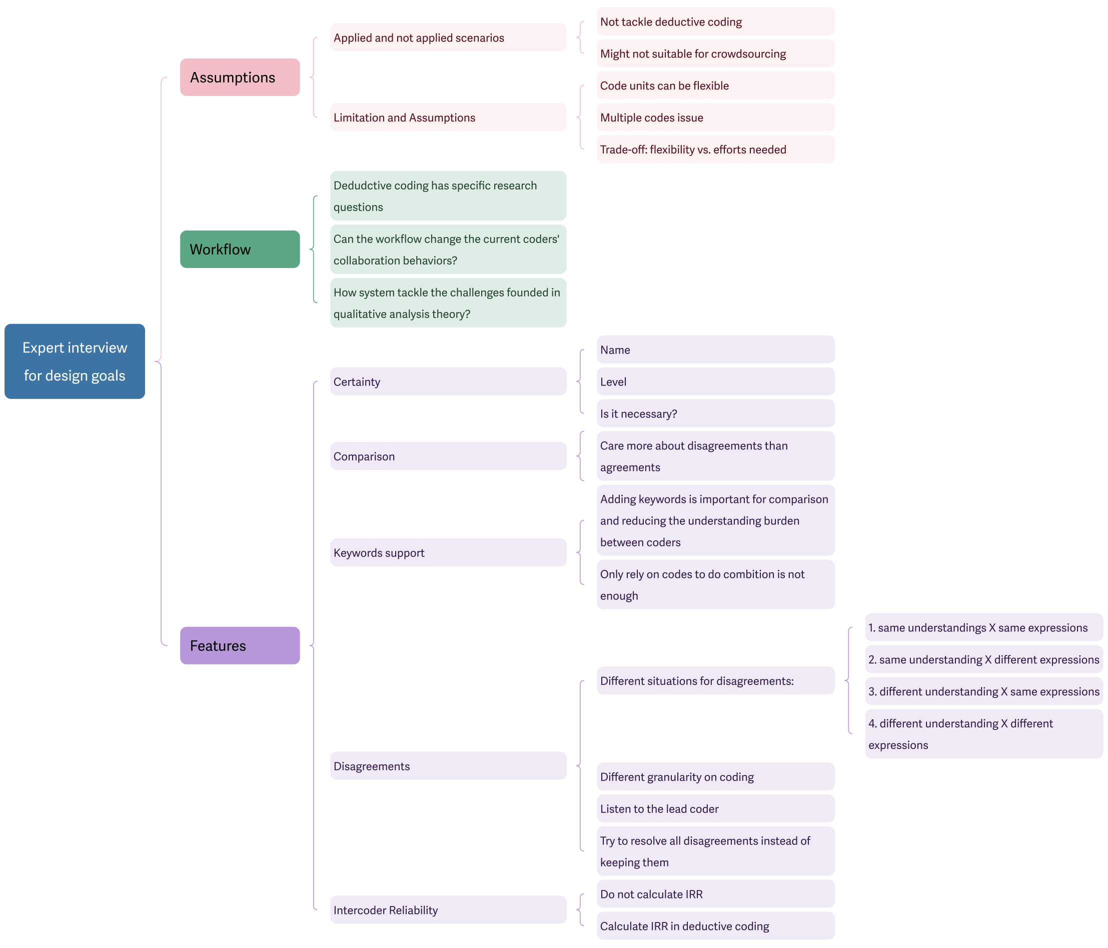

Expert interview for design goals
Assumptions
Applied and not applied scenarios
Not tackle deductive coding
Might not suitable for crowdsourcing
Limitation and Assumptions
Code units can be flexible
Multiple codes issue
Trade-off: flexibility vs. efforts needed
Workflow
Dedudctive coding has specific research questions
Can the workflow change the current coders' collaboration behaviors?
How system tackle the challenges founded in qualitative analysis theory?
Features
Certainty
Name
Level
Is it necessary?
Comparison
Care more about disagreements than agreements
Keywords support
Adding keywords is important for comparison and reducing the understanding burden between coders
Only rely on codes to do combition is not enough
Disagreements
Different situations for disagreements:
1. same understandings X same expressions
2. same understanding X different expressions
3. different understanding X same expressions
4. different understanding X different expressions
Different granularity on coding
Listen to the lead coder
Try to resolve all disagreements instead of keeping them
Intercoder Reliability
Do not calculate IRR
Calculate IRR in deductive coding

FIGURE B.4: Results of thematic analysis in Step3 (Section 6.2.1) from expert interviews to derive design goals, with each node representing a coded element.

TABLE B.2: The prompts utilized in CollabCoder in Phase 1 when communicating with the ChatGPT API to produce code suggestions for text.

Phases	Features	Prompt Template	Example
Phase 1	Seek code suggestions for units	• **system role**: You are a helpful qualitative analysis assistant, aiding researchers in developing codes that can be utilized in subsequent stages, including discussions for creating codebooks and final coding processes; • **user input**: Please create three general summaries for [text] (within six-word);	[Text]: "How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters." **Three general summaries for the above [Text]**: 1. Book enlightened my initial business journey. 2. Comprehensive guide for business undergraduates. 3. Knowledge boost for new business students.
	Seek most relevant codes from coding history	• **system role**: You are a helpful qualitative analysis assistant, aiding researchers in developing codes that can be utilized in subsequent stages, including discussions for creating codebooks and final coding processes; • **user input**: Please identify the top three codes relevant to this [text] from the following [Code list]; 1. [Code] 2. [Code] ... Here is the example format of results: 1. code content 2. code content 3. code content	[Text] "How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters." [Code list] 1. Detailed introduction to business relations 2. Inspiring guide to improve life 3. Journey of light and love. 4. Easy to read, highlight-worthy 5. Well-written lesson on simplicity 6. Rodriguez tells truth, Pelosi lies **Three relevant codes to [Text] from [Code list]**: 1. Detailed introduction to business relations 2. Easy to read, highlight-worthy. 3. Well-written lesson on simplicity.

TABLE B.3: The prompts utilized in CollabCoder in Phase 2 when communicating with the ChatGPT API to produce code suggestions for text.

Phases	Features	Prompt Template	Example
Phase 2	Make code decisions	• **system role**: You are a helpful qualitative analysis assistant, aiding researchers in developing final codes that can be utilized in subsequent stages, including final coding processes; • **user input**: Please create three concise, non-repetitive, and general six-word code combinations for the [text] using [Code1] and [Code2]: 1. [Code] 2. [Code] ... Requirements: 1. 6 words or fewer; 2. No duplicate words; 3. Be general; 4. Three distinct versions Here is the format of results: Version1: code content Version2: code content Version3: code content	[Text] "How A Business Works was an excellent book to read as I began my first semester as a college student. Although my goal is to major in Business, I started my semester off with no idea of even the basic guidelines a Business undergrad should know. This book describes in detail every aspect dealing with business relations, and I enjoyed reading it. It felt great going to my additional business classes prepared and knowledgeable on the subject they were describing. Very well written, Professor Haeberle! I recommend this book to anyone and everyone who would like additional knowledge pertaining to business matters." [Code1]: Detailed introduction to business relations. [Code2]: Comprehensive guide to business basics **Three suggestions for final codes:** Version1: In-depth overview of business fundamentals Version2: Thorough guide to business relationships Version3: Comprehensive resource on business essentials

TABLE B.4: The prompts utilized in CollabCoder in Phase 3 when communicating with the ChatGPT API to produce code group suggestions for final code decisions.

Phases	Features	Prompt Template	Example
Phase 3	Generate code groups	• **system role**: You are a helpful qualitative analysis assistant, aiding researchers in generating final code groups/main themes based on the [Code list] provided, in order to give an overview of the main content of the coding. • **user input**: Organize the following [Code list] into 3 thematic groups without altering the original codes, and name each group: 1. [Code] 2. [Code] ... Here is the format of the results: Group1: [theme] 1.[code] 2.[code] 3.[code]	[Code list]: 1. Simplified business knowledge 2. Unconventional, but valuable business insights. 3. Effective lessons on simplicity 4. Innovative leadership through Jugaad. 5. Cautionary book on costly Google campaigns. 6. Timeless love principles improve business. 7. Politicians deceive for political gain. 8. A high school must-read for financial literacy. 9. Entertaining and educational graphic novel. **Three code groups for the above [Code list]**: Group1: Simplified business knowledge 1. Simplified business knowledge 2. Effective lessons on simplicity 3. Cautionary book on costly Google campaigns. Group2: Inspiring and practical personal development book 1. Timeless love principles improve business. 2. A high school must-read for financial literacy. 3. Entertaining and educational graphic novel. Group3: Unconventional, but valuable business insights 1. Innovative leadership through Jugaad. 2. Unconventional, but valuable business insights. 3. Politicians deceive for political gain.

TABLE B.5: Demographics of Participants in User Evaluation. Note: QA expertise is not solely determined by the number of QA experiences, but also by the level of QA knowledge. This is why some participants with 1-3 times of prior experience may still regard themselves as having intermediate expertise.

Pairs		**English**	**Job**	**Education**	**Related experience**	**Self-reported QA expertise**	**QA times**	**Software for QA**
Pair 1	P1	Proficient	Student	Master	Basic understanding of qualitative research method	No Experience	None	None
	P2	First language	Automation QA Engineer	Undergraduate	Automation	No Experience	None	None
Pair 2	P3	First language	Phd Student	PhD and above	HCI	Expert	7 times above	Atlas.ti Desktop
	P4	First language	Undergraduate	Undergraduate	Business analytics with Python and R	No Experience	None	None
Pair 3	P5	Proficient	Student	Undergraduate	Coding with Python	Beginner	1-3 times	None
	P6	First language	Research Assistant	Master	Asian studies	Expert	7 times above (mainly interview data)	Word, Excel, Dedoose
Pair 4	P7	First language	Data Analyst	Undergraduate	Data Visualisation	No Experience	None	None
	P8	First language	Student	Undergraduate	R, HTML/CSS, Market research	Beginner	1-3 times	R
Pair 5	P9	First language	Research assistant	Undergraduate	Learning science, Grounded theory	Intermediate	4-6 times	nVivo
	P10	First language	Data science intern	Undergraduate	Computer Vision, Python	No Experience	None	None
Pair 6	P11	First language	Behavioral Scientist	Undergraduate	Psychology, Behavioral Science, Thematic analysis	Intermediate	1-3 times	Word
	P12	First language	Student	Undergraduate	Accounting & Python, SQL	No experience	None	None
Pair 7	P13	First language	Research Assistant	Undergraduate	SPSS, Python, basic qualitative analysis understanding, topic modeling	Beginner	1-3 times	None
	P14	First language	Research Assistant	Undergraduate	Have research experience using QA for interview transcription	Intermediate	7 times above	nVivo, Excel
Pair 8	P15	First language	Researcher	Master	Thematic analysis for interview, literature review	Beginner	1-3 times	fQCA
	P16	First language	Student	Master	Social science	No experience	None	None

TABLE B.6: Observation notes for Pair1-Pair4. The language has been revised for readability.

	Atlas.ti Web	CollabCoder
P1xP2	Even individuals familiar with Google Docs/Excel might find it challenging to adapt to Atlas.ti, P1's learning pace was even slower than P2's. P1's coding was more detailed and extensive than P2's, making his codes longer and more content-rich. His lack of experience in ML further hampered his speed, causing a significant delay in the coding process. In a 30-minute span, he managed only 5 codes compared to another coder who completed all 20. Due to time constraints, we had to stop the process. Over time, the involvement of the other coder diminished. Only one coder, more adept with the platform, primarily handled the coding process. The other coder, like P1, shifted to a supportive role, offering input on the final report and the categorization phase.	This time around, P1 found it easier to start coding. Both he and the other coder seldom used the "Similar codes" function. Additionally, they rarely used the "certainty" button, indicating a potential issue of over-reliance on certain features or methods.
P3xP4	Even the expert coder (P3) faced challenges learning the software and initially felt lost navigating its interface. Additionally, she found it difficult to identify the origin of selected text when only a portion is chosen from the original unit. In both coding sessions, the lead coder shares her screen and invites the other coder to offer suggestions for combining codes and arriving at the final code. Due to the limitations of the software, they are obliged to manually search for similar codes, relying on visual inspection to group them together.	P3 is a conscientious coder who is concerned about potentially slowing down the overall pace of the study. To address this, she intermittently checks the progress of others to adjust her own workflow. She finds this feature to be "quite helpful." When it comes to coding, if the codes are identical, they typically don't consult definitions. While ChatGPT serves as a reference point for decision-making, it is not strictly followed. If there's a difference in understanding, they will refer to ChatGPT for final decision support. When P4 sets the certainty level to 2, it signifies "I'm not sure what this person is talking about." The lead coder is conscious of not overly relying on her own codes, as she doesn't want to appear too dominant within the team. By using third-party codes, she aims to maintain a more balanced influence. P4 also mentioned that he sometimes assigns low priority to definitions because he has only a few to refer to. During the decision-making process, direct selections from ChatGPT suggestions have become less frequent. Instead, the team tends to use ChatGPT is more of a point of reference. This seems to indicate that they are becoming more cautious in their approach.
P5xP6	Beginners have the option of referring to others' codes as a starting point for their own coding endeavors. P6, for instance, prefers to check the code history. This approach can provide valuable insights and context, helping new coders understand the coding process better and potentially speeding up their learning curve. The team takes advantage of the auto-completion function, typing in just a few words and then clicking the check button to select existing codes instead of creating new ones. When P5 is coding, he initially refers to other people's codes before adding his own. The codes generated by both coders tend to be rather general. They often refer to each other's work, with the beginner usually following the coding scheme established by the more experienced coder.	Russell is not familiar with the new coding method and initially selected the entire text as "keywords support", thinking that only the selected portion would be coded. This suggests that users may need some training to effectively use the coding system.
P7xP8	To speed up the coding process, only one coder takes on the responsibility of combining and grouping the codes into thematic clusters.	P7 and P8 both tend to use ChatGPT sparingly, favoring the creation of their own codes. P7 mentions that the long latency for ChatGPT's suggestions is a factor; if the results aren't quick, he opts to input his own codes. P8 notes that she often has to edit ChatGPT's suggestions, deleting some words to better tailor them to her needs. However, they are more likely to choose suggestions from ChatGPT if they want to expedite the process. Even if they don't ultimately select a ChatGPT suggestion, they still refer to these codes as a reference point. This approach aligns with the sentiment that AI can't be trusted for every task; it serves as a tool rather than a definitive authority. If there's any uncertainty about why a code is part of a specific group, or if the meaning of a code within a group is unclear, they will refer back to the original text during the code grouping phase for clarification. By highlighting keywords and listing them, the coders are able to work asynchronously instead of in real-time. This approach allows each coder to leave markers of their understanding, facilitating a smoother integration of perspectives without the need for immediate discussion.

TABLE B.7: Observation notes for Pair5-Pair8. The language has been revised for readability.

	Atlas.ti Web	**CollabCoder**
P9xP10	Normal collaboration process, no specific notes	The coding process involves multiple steps: initially reading the data, requesting suggestions, reviewing those suggestions, returning to the raw data for another check, and then choosing or editing the code. After this, keywords are added and the level of certainty is labeled. When it comes to merging codes, the team starts with a preliminary idea for a final decision, and then consults ChatGPT to generate a final, merged code.
P11xP12	P12 adopts a strategy of starting his coding from the last data point and working his way to the top, in an effort to minimize overlap and influence from P11. The pace at which each coder works varies significantly: one coder is much faster than the other and, consequently, contributes more to the overall workload. In time-sensitive situations, the quicker coder naturally takes on more responsibilities than the slower coder. This dynamic could have implications for the diversity and depth of coding, as the faster coder's perspectives may disproportionately influence the final output.	A less-than-ideal scenario for discussion. The team may overly rely on ChatGPT's generated decisions due to time constraints. In these cases, substantive discussion is replaced with shortcuts like simply choosing "the first one" or "the second one" from ChatGPT's suggestions. This is a notable drawback for the research, as it sidesteps deeper analysis and thought, leading to concerns about over-reliance on automated suggestions. The dynamic often results in the more experienced coder taking a dominant role in the process, which could impact the diversity of perspectives in the coding. AI does offer the advantage of allowing users to work with longer text segments compared to manual coding, which often focuses on keywords or smaller data units. However, this advantage should not come at the expense of thoughtful discussion and careful consideration in the coding process.
P13xP14	Normal collaboration process, no specific notes	The overall coding process appears to be smooth. Both coders generally agree and neither is overly dominant; they respect each other's opinions. Because of this harmony, they utilize ChatGPT to generate the final code expressions. It seems that AI plays a significant role in the code grouping process, directing the way codes are organized. When the coders are working independently, they tend to group codes based on sentiment analysis. However, under AI's guidance, their focus shifts to content analysis. This suggests that while AI can be a helpful tool, its influence can also steer the analytical direction, which may or may not align with the coders' initial approach or intentions.
P15xP16	Due to time constraints, discussions between the coders are notably brief and to the point. There isn't much room for extended dialogue or deeper analysis, which could have implications for the thoroughness and quality of the coding process.	Participants generally start by reading the original text, then request suggestions from ChatGPT before proceeding to code the data. P16 follows this sequence, reading the text first and then consulting ChatGPT's suggestions. P15, on the other hand, sometimes deletes her initial code entry to generate a different version. Due to time constraints, she doesn't delve deeply into the text and may even skip over some sections. To save time, she might initiate ChatGPT's code suggestion process for another text segment while working on the current one. Both P15 and P16 demonstrate mutual respect when their codes closely align (with a similarity score greater than 0.9). They don't particularly mind whose code is used for the final decision. For instance, they may choose one of P15's codes for one text segment and switch to another code from P16 for a different segment. If their coding doesn't match despite having similar evidence, they discuss the reasons behind their code choices. The coder with the more explainable rationale usually wins out, with the other coder simply saying, "Use yours." If they can't reach a decision, they turn to ChatGPT for a suggested code. When neither coder feels confident in their understanding of the raw data, they'll admit their uncertainty, often stating, "I have no idea about this", before potentially seeking further guidance.

Bibliography

Akpınar, Ercan, Demet Erol, and Bülent Aydoğdu (2009). "The role of cognitive conflict in constructivist theory: An implementation aimed at science teachers". In: *Procedia - Social and Behavioral Sciences* 1.1. World Conference on Educational Sciences: New Trends and Issues in Educational Sciences, pp. 2402–2407. ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2009.01.421. URL: https://www.sciencedirect.com/science/article/pii/S1877042809004248.

Allen, J.E., C.I. Guinn, and E. Horvtz (1999). "Mixed-initiative interaction". In: *IEEE Intelligent Systems and their Applications* 14.5, pp. 14–23. DOI: 10.1109/5254.796083.

Altman, Douglas G and J Martin Bland (1995). "Statistics notes: Absence of evidence is not evidence of absence". In: *Bmj* 311.7003, p. 485. DOI: https://doi.org/10.1136/bmj.311.7003.485.

Amershi, Saleema et al. (2014). "Power to the people: The role of humans in interactive machine learning". In: *Ai Magazine* 35.4, pp. 105–120.

Amiryousefi, Mohammad et al. (2021). "Impact of Etherpad-based Collaborative Writing Instruction on EFL Learners' Writing Performance, Writing Self-efficacy, and Attribution: A Mixed-Method Approach". In: *Two Quarterly Journal of English Language Teaching and Learning University of Tabriz* 13.28, pp. 19–37.

Anderson, Ross C, Meg Guerreiro, and Joanna Smith (2016). "Are all biases bad? Collaborative grounded theory in developmental evaluation of education policy". In: *Journal of Multidisciplinary Evaluation* 12.27, pp. 44–57. DOI: https://doi.org/10.56645/jmde.v12i27.449.

Ashktorab, Zahra, Michael Desmond, et al. (2021). "AI-Assisted Human Labeling: Batching for Efficiency without Overreliance". In: *Proceedings of the ACM on Human-Computer Interaction* 5.CSCW1, pp. 1–27.

Ashktorab, Zahra, Q. Vera Liao, et al. (Oct. 2020). "Human-AI Collaboration in a Cooperative Game Setting: Measuring Social Perception and Outcomes". In: *Proc. ACM Hum.-Comput. Interact.* 4.CSCW2. DOI: 10.1145/3415167. URL: https://doi.org/10.1145/3415167.

Azungah, Theophilus (2018). "Qualitative research: deductive and inductive approaches to data analysis". In: *Qualitative research journal* 18.4, pp. 383–400.

Bach, Tita Alissa et al. (2022). "A Systematic Literature Review of User Trust in AI-Enabled Systems: An HCI Perspective". In: *International Journal of Human–Computer Interaction*, pp. 1–16.

BANOVIC, NIKOLA et al. (2023). "Being Trustworthy is Not Enough: How Untrustworthy Artificial Intelligence (AI) Can Deceive the End-Users and Gain Their Trust". In.

Barry, Christine A. et al. (1999). "Using Reflexivity to Optimize Teamwork in Qualitative Research". In: *Qualitative Health Research* 9.1. PMID: 10558357, pp. 26–44. DOI: 10.1177/104973299129121677. eprint: https://doi.org/10.1177/104973299129121677. URL: https://doi.org/10.1177/104973299129121677.

Baumer, Eric PS et al. (2017). "Comparing grounded theory and topic modeling: Extreme divergence or unlikely convergence?" In: *Journal of the Association for Information Science and Technology* 68.6, pp. 1397–1410.

Bebermeier, Sarah and Denise Kerkhoff (2019). "Use and Impact of the Open Source Online Editor Etherpad in a Psychology Students' Statistics Class." In: *Psychology Teaching Review* 25.2, pp. 30–38.

Bjørn, Pernille et al. (Nov. 2014). "Does Distance Still Matter? Revisiting the CSCW Fundamentals on Distributed Collaboration". In: *ACM Trans. Comput.-Hum. Interact.* 21.5. ISSN: 1073-0516. DOI: 10.1145/2670534. URL: https://doi.org/10.1145/2670534.

Bocklisch, Tom et al. (2017). "Rasa: Open source language understanding and dialogue management". In: *arXiv preprint arXiv:1712.05181*.

Brachman, Michelle et al. (2022). "Reliance and Automation for Human-AI Collaborative Data Labeling Conflict Resolution". In: *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–27.

Bradley, Jana (1993). "Methodological issues and practices in qualitative research". In: *The Library Quarterly* 63.4, pp. 431–449.

Braun, Virginia and Victoria Clarke (2006). "Using thematic analysis in psychology". In: *Qualitative research in psychology* 3.2, pp. 77–101. DOI: DOI:10.1191/1478088706qp063oa.

Bryant, Antony and Kathy Charmaz (2007). *The Sage handbook of grounded theory*. Sage.

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

Bunk, Tanja et al. (2020). "Diet: Lightweight language understanding for dialogue systems". In: *arXiv preprint arXiv:2004.09936*.

Byun, Courtni, Piper Vasicek, and Kevin Seppi (2023). "Dispensing with Humans in Human-Computer Interaction Research". In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI EA '23. Hamburg, Germany: Association for Computing Machinery. ISBN: 9781450394222. DOI: 10.1145/3544549.3582749. URL: https://doi.org/10.1145/3544549.3582749.

Campero, Andres et al. (2022). "A test for evaluating performance in human-computer systems". In: *arXiv preprint arXiv:2206.12390*.

Cao, Junming et al. (2023). *Understanding the Complexity and Its Impact on Testing in ML-Enabled Systems*. arXiv: 2301.03837 [cs.SE].

Cao, Shiye and Chien-Ming Huang (2022). "Understanding User Reliance on AI in Assisted Decision-Making". In: *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–23.

Cash, Philip J. (2018). "Developing theory-driven design research". In: *Design Studies* 56, pp. 84–119. ISSN: 0142-694X. DOI: https://doi.org/10.1016/j.destud.2018.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0142694X18300140.

Ceccato, Mariano et al. (2004). "Ambiguity identification and measurement in natural language texts". In.

Charmaz, Kathy (2014). *Constructing grounded theory*. sage.

Chen, Li et al. (Oct. 2013). "Human Decision Making and Recommender Systems". In: *ACM Trans. Interact. Intell. Syst.* 3.3. ISSN: 2160-6455. DOI: 10.1145/2533670.2533675. URL: https://doi.org/10.1145/2533670.2533675.

Chen, Nan-Chen et al. (June 2018). "Using Machine Learning to Support Qualitative Coding in Social Science: Shifting the Focus to Ambiguity". In: *ACM Trans. Interact. Intell. Syst.* 8.2. ISSN: 2160-6455. DOI: 10.1145/3185515. URL: https://doi.org/10.1145/3185515.

Chen, Nan-chen et al. (2016). "Challenges of applying machine learning to qualitative coding". In: *ACM SIGCHI Workshop on Human-Centered Machine Learning*.

Cheng, Hao-Fei et al. (2019). "Explaining Decision-Making Algorithms through UI: Strategies to Help Non-Expert Stakeholders". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, pp. 1–12. ISBN: 9781450359702. DOI: 10.1145/3290605.3300789. URL: https://doi.org/10.1145/3290605.3300789.

Chinh, Bonnie et al. (2019). "Ways of Qualitative Coding: A Case Study of Four Strategies for Resolving Disagreements". In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI EA '19. Glasgow, Scotland Uk: Association for Computing Machinery, pp. 1–6. ISBN: 9781450359719. DOI: 10.1145/3290607.3312879. URL: https://doi.org/10.1145/3290607.3312879.

Clark, Herbert H. and Susan E. Brennan (1991). "Grounding in Communication". In: *Perspectives on Socially Shared Cognition*. Ed. by Lauren Resnick et al. American Psychological Association, pp. 13–1991. DOI: https://doi.org/10.1037/10096-006.

Collins, Christopher S. and Carrie M. Stockton (2018). "The Central Role of Theory in Qualitative Research". In: *International Journal of Qualitative Methods* 17.1, p. 1609406918797475.

DOI: 10.1177/1609406918797475. eprint: https://doi.org/10.1177/1609406918797475. URL: https://doi.org/10.1177/1609406918797475.

Corbin, Juliet and Anselm Strauss (2008). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage publications Sage.

— (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

Corbin, Juliet M and Anselm Strauss (1990). "Grounded theory research: Procedures, canons, and evaluative criteria". In: *Qualitative sociology* 13.1, pp. 3–21. DOI: https://doi.org/10.1007/BF00988593.

Cornish, Flora, Alex Gillespie, and Tania Zittoun (2013). "Collaborative analysis of qualitative data". In: *The SAGE handbook of qualitative data analysis* 79, p. 93. DOI: https://doi.org/10.4135/9781446282243.

Crowston, Kevin, Xiaozhong Liu, and Eileen E Allen (2010). "Machine learning and rule-based automated coding of qualitative data". In: *proceedings of the American Society for Information Science and Technology* 47.1, pp. 1–2.

Davani, Aida Mostafazadeh, Mark Díaz, and Vinodkumar Prabhakaran (2022). "Dealing with Disagreements: Looking Beyond the Majority Vote in Subjective Annotations". In: *Transactions of the Association for Computational Linguistics* 10, pp. 92–110. DOI: 10.1162/tacl_a_00449. URL: https://aclanthology.org/2022.tacl-1.6.

DeCuir-Gunby, Jessica T, Patricia L Marshall, and Allison W McCulloch (2011). "Developing and using a codebook for the analysis of interview data: An example from a professional development research project". In: *Field methods* 23.2, pp. 136–155.

Díaz, Jessica et al. (2023). "Applying Inter-Rater Reliability and Agreement in collaborative Grounded Theory studies in software engineering". In: *Journal of Systems and Software* 195, p. 111520. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111520. URL: https://www.sciencedirect.com/science/article/pii/S0164121222001960.

Drouhard, Margaret et al. (2017). "Aeonium: Visual analytics to support collaborative qualitative coding". In: *2017 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, pp. 220–229.

Dzindolet, Mary T et al. (2003). "The role of trust in automation reliance". In: *International journal of human-computer studies* 58.6, pp. 697–718.

Emamgholizadeh, Hanif (2022). "Supporting Group Decision-Making Processes Based on Group Dynamics". In: *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*. UMAP '22. Barcelona, Spain: Association for Computing Machinery, pp. 346–350. ISBN: 9781450392075. DOI: 10.1145/3503252.3534358. URL: https://doi.org/10.1145/3503252.3534358.

Entin, Elliot E. (2000). “Performance and Process Measure Relationships in Transitioning from a Low to High Fidelity Simulation Environment”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 44.1, pp. 280–280. DOI: 10.1177/154193120004400177. eprint: https://doi.org/10.1177/154193120004400177. URL: https://doi.org/10.1177/154193120004400177.

Eyuboglu, Sabri et al. (2022). “dcbench: a benchmark for data-centric AI systems”. In: *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, pp. 1–4.

Farra, Noura et al. (2010). “Sentence-level and document-level sentiment mining for arabic texts”. In: *2010 IEEE international conference on data mining workshops*. IEEE, pp. 1114–1119.

Felix, Cristian, Aritra Dasgupta, and Enrico Bertini (2018). “The exploratory labeling assistant: Mixed-initiative label curation with large document collections”. In: *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pp. 153–164.

Fereday, Jennifer and Eimear Muir-Cochrane (2006). “Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development”. In: *International journal of qualitative methods* 5.1, pp. 80–92.

Feuston, Jessica L. and Jed R. Brubaker (Oct. 2021). “Putting Tools in Their Place: The Role of Time and Perspective in Human-AI Collaboration for Qualitative Analysis”. In: *Proc. ACM Hum.-Comput. Interact.* 5.CSCW2. DOI: 10.1145/3479856. URL: https://doi.org/10.1145/3479856.

Flick, Uwe (2013). *The SAGE handbook of qualitative data analysis*. Sage.

Freitas, Fábio et al. (2017). “Learn for yourself: The self-learning tools for qualitative analysis software packages”. In: *Digital Education Review* 32, pp. 97–117.

Ganji, Abbas, Mania Orand, and David W. McDonald (Nov. 2018). “Ease on Down the Code: Complex Collaborative Qualitative Coding Simplified with ’Code Wizard’”. In: *Proc. ACM Hum.-Comput. Interact.* 2.CSCW. DOI: 10.1145/3274401. URL: https://doi.org/10.1145/3274401.

Gao, Jie, Kenny Tsu Wei Choo, et al. (Aug. 2023). “CoAIcoder: Examining the Effectiveness of AI-Assisted Human-to-Human Collaboration in Qualitative Analysis”. In: *ACM Trans. Comput.-Hum. Interact.* Just Accepted. ISSN: 1073-0516. DOI: 10.1145/3617362. URL: https://doi.org/10.1145/3617362.

Gao, Jie, Yuchen Guo, Toby Jia-Jun Li, et al. (2023). “CollabCoder: A GPT-Powered WorkFlow for Collaborative Qualitative Analysis”. In: *Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing*. CSCW ’23 Companion. Minneapolis, MN, USA: Association for Computing Machinery, pp. 354–357. ISBN: 9798400701290. DOI: 10.1145/3584931.3607500. URL: https://doi.org/10.1145/3584931.3607500.

Gao, Jie, Yuchen Guo, Gionnieve Lim, et al. (2023). *CollabCoder: A GPT-Powered Workflow for Collaborative Qualitative Analysis*. arXiv: 2304.07366 [cs.HC].

Gao, Jie, Leijing Zhou, et al. (2018). "Expressive Plant: A Multisensory Interactive System for Sensory Training of Children with Autism". In: *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. UbiComp '18. Singapore, Singapore: Association for Computing Machinery, pp. 46–49. ISBN: 9781450359665. DOI: 10.1145/3267305.3267588. URL: https://doi.org/10.1145/3267305.3267588.

Gebreegziabher, Simret Araya et al. (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. Hamburg, Germany: Association for Computing Machinery. ISBN: 9781450394215. DOI: 10.1145/3544548.3581352. URL: https://doi.org/10.1145/3544548.3581352.

Giraud-Carrier, Christophe (Dec. 2000). "A Note on the Utility of Incremental Learning". In: *AI Commun.* 13.4, pp. 215–223. ISSN: 0921-7126.

Glaser, Barney and Anselm Strauss (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.

Golafshani, Nahid (2003). "Understanding reliability and validity in qualitative research". In: *The qualitative report* 8.4, pp. 597–607. DOI: https://doi.org/10.46743/2160-3715/2003.1870.

Goldman, Max, Greg Little, and Robert C Miller (2011). "Real-time collaborative coding in a web IDE". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 155–164.

Grad Coach (2023). *Qualitative Data Analysis Methods: Top 6 + Examples*. https://gradcoach.com/qualitative-data-analysis-methods/.

Grimes, G. Mark, Ryan M. Schuetzler, and Justin Scott Giboney (2021). "Mental models and expectation violations in conversational AI interactions". In: *Decision Support Systems* 144, p. 113515. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2021.113515. URL: https://www.sciencedirect.com/science/article/pii/S0167923621000257.

Hackshaw, Allan (2008). *Small studies: strengths and limitations*. DOI: https://doi.org/10.1183/09031936.00136408.

Hall, Wendy A et al. (2005). "Qualitative teamwork issues and strategies: Coordination through mutual adjustment". In: *Qualitative Health Research* 15.3, pp. 394–410.

Hämäläinen, Perttu, Mikke Tavast, and Anton Kunnari (2023). "Evaluating Large Language Models in Generating Synthetic HCI Research Data: A Case Study". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. <conf-loc>, <city>Hamburg</city>, <country>Germany</country>, </conf-loc>: Association for Computing Machinery. ISBN: 9781450394215. DOI: 10.1145/3544548.3580688. URL: https://doi.org/10.1145/3544548.3580688.

Hong, Matt-Heun et al. (2022). "Scholastic: Graphical Human-AI Collaboration for Inductive and Interpretive Text Analysis". In: *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. UIST '22. Bend, OR, USA: Association for Computing Machinery. ISBN: 9781450393201. DOI: 10.1145/3526113.3545681. URL: https://doi.org/10.1145/3526113.3545681.

Hopper, Tim et al. (2021). "YouTube for transcribing and Google drive for collaborative coding: Cost-effective tools for collecting and analyzing interview data". In: *The Qualitative Report* 26.3, pp. 861–873. DOI: https://doi.org/10.46743/2160-3715/2021.4639.

Horvitz, Eric (1999). "Principles of Mixed-Initiative User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, pp. 159–166. ISBN: 0201485591. DOI: 10.1145/302979.303030. URL: https://doi.org/10.1145/302979.303030.

Interaction Institute for Social Change (July 2018). *Power Dynamics: The Hidden Element to Effective Meetings*. https://interactioninstitute.org/power-dynamics-the-hidden-element-to-effective-meetings/.

Ippolito, Daphne et al. (2022). *Creative Writing with an AI-Powered Writing Assistant: Perspectives from Professional Writers*. arXiv: 2211.05030 [cs.HC].

Jameson, Anthony, Stephan Baldes, and Thomas Kleinbauer (2003). "Enhancing mutual awareness in group recommender systems". In: *Proceedings of the IJCAI*. Vol. 10. 989863.989948.

Janis, Irving L (2008). "Groupthink". In: *IEEE Engineering Management Review* 36.1, p. 36. DOI: 10.1109/EMR.2008.4490137.

Jiang, Jialun Aaron et al. (Apr. 2021). "Supporting Serendipity: Opportunities and Challenges for Human-AI Collaboration in Qualitative Analysis". In: *Proc. ACM Hum.-Comput. Interact.* 5.CSCW1. DOI: 10.1145/3449168. URL: https://doi.org/10.1145/3449168.

Jörg Hecker, Neringa Kalpokas (2023). *The Ultimate Guide to Qualitative Research - Part 1: The Basics*. URL: https://atlasti.com/guides/qualitative-research-guide-part-1/theoretical-perspective.

Kaufmann, Andreas, Ann Barcomb, and Dirk Riehle (2020). "Supporting Interview Analysis with Autocoding." In: *HICSS*, pp. 1–10.

Kim, Tae Soo et al. (2023). "Cells, generators, and lenses: Design framework for object-oriented interaction with large language models". In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–18.

Knowles, Bran et al. (2015). “Models and patterns of trust”. In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pp. 328–338.

Kocielnik, Rafal, Saleema Amershi, and Paul N Bennett (2019). “Will you accept an imperfect ai? exploring designs for adjusting end-user expectations of ai systems”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14.

Kok, Ties de (2023). “Generative LLMs and Textual Analysis in Accounting:(Chat) GPT as Research Assistant?” In: *Available at SSRN*.

Kurasaki, Karen S (2000). “Intercoder reliability for validating conclusions drawn from open-ended interview data”. In: *Field methods* 12.3, pp. 179–194. DOI: https://doi.org/10.1177/1525822X0001200301.

Lazar, Jonathan, Jinjuan Heidi Feng, and Harry Hochheiser (May 2017a). *Research Methods in Human-Computer Interaction*. English. 2nd edition. Cambridge, MA: Morgan Kaufmann. ISBN: 978-0-12-805390-4.

— (2017b). *Research methods in human-computer interaction*. Morgan Kaufmann.

Lee, John D and Katrina A See (2004). “Trust in automation: Designing for appropriate reliance”. In: *Human factors* 46.1, pp. 50–80. DOI: https://doi.org/10.1518/hfes.46.1.50_30392.

Lee, Mina, Percy Liang, and Qian Yang (2022). “CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities”. In: *arXiv preprint arXiv:2201.06796*.

Leeson, William et al. (2019). “Natural Language Processing (NLP) in qualitative public health research: a proof of concept study”. In: *International Journal of Qualitative Methods* 18, p. 1609406919887021.

Li, Mengxiang et al. (2013). “Helpfulness of online product reviews as seen by consumers: Source and content features”. In: *International Journal of Electronic Commerce* 17.4, pp. 101–136.

Liao, Q. Vera, Daniel Gruen, and Sarah Miller (2020). “Questioning the AI: Informing Design Practices for Explainable AI User Experiences”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI ’20. Honolulu, HI, USA: Association for Computing Machinery, pp. 1–15. ISBN: 9781450367080. DOI: 10.1145/3313831.3376590. URL: https://doi.org/10.1145/3313831.3376590.

Liew, Jasy Suet Yan et al. (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*, pp. 44–48.

Lindgren, Britt-Marie, Berit Lundman, and Ulla H Graneheim (2020). “Abstraction and interpretation during the qualitative content analysis process”. In: *International journal of nursing studies* 108, p. 103632.

Liu, Huiting et al. (2022). “Model Stability with Continuous Data Updates”. In: *CoRR* abs/2201.05692. arXiv: 2201.05692. URL: https://arxiv.org/abs/2201.05692.

Liu, Pengfei et al. (2023). “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing”. In: *ACM Computing Surveys* 55.9, pp. 1–35.

Lubars, Brian and Chenhao Tan (2019). “Ask Not What AI Can Do, But What AI Should Do: Towards a Framework of Task Delegability”. In: *CoRR* abs/1902.03245. arXiv: 1902.03245. URL: http://arxiv.org/abs/1902.03245.

Maguire, Moira and Brid Delahunt (2017). “Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars.” In: *All Ireland Journal of Higher Education* 9.3.

Maher, Carmel et al. (2018). “Ensuring rigor in qualitative data analysis: A design research approach to coding combining NVivo with traditional material methods”. In: *International journal of qualitative methods* 17.1, p. 1609406918786362.

Malone, Thomas W. and Kevin Crowston (Mar. 1994). “The Interdisciplinary Study of Coordination”. In: *ACM Comput. Surv.* 26.1, pp. 87–119. ISSN: 0360-0300. DOI: 10.1145/174666.174668. URL: https://doi.org/10.1145/174666.174668.

Mäntylä, Mika V et al. (2015). “On rapid releases and software testing: a case study and a semi-systematic literature review”. In: *Empirical Software Engineering* 20, pp. 1384–1425.

Marathe, Megh and Kentaro Toyama (2018a). “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*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, pp. 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173922. URL: https://doi.org/10.1145/3173574.3173922.

— (2018b). “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*, pp. 1–12.

Mayer, Roger C, James H Davis, and F David Schoorman (1995). “An integrative model of organizational trust”. In: *Academy of management review* 20.3, pp. 709–734.

McDonald, Nora, Sarita Schoenebeck, and Andrea Forte (Nov. 2019). “Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice”. In: *Proc. ACM Hum.-Comput. Interact.* 3.CSCW. DOI: 10.1145/3359174. URL: https://doi.org/10.1145/3359174.

McHugh, Mary L (2012). “Interrater reliability: the kappa statistic”. In: *Biochemia medica* 22.3, pp. 276–282.

Metrics (n.d.). URL: https://darel13712.github.io/rs_metrics/metrics/.

Moravcsik, Andrew (2014). “Transparency: The Revolution in Qualitative Research”. In: *PS: Political Science; Politics* 47.1, pp. 48–53. DOI: 10.1017/S1049096513001789.

Motamedi, Mohammad, Nikolay Sakharnykh, and Tim Kaldewey (2021). "A data-centric approach for training deep neural networks with less data". In: *arXiv preprint arXiv:2110.03613*.

Muller, Michael et al. (2016). "Machine learning and grounded theory method: convergence, divergence, and combination". In: *Proceedings of the 19th international conference on supporting group work*, pp. 3–8.

Naim, Iftekhar et al. (2015). "Automated prediction and analysis of job interview performance: The role of what you say and how you say it". In: *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. Vol. 1, pp. 1–6. DOI: 10.1109/FG.2015.7163127.

Nassaji, Hossein (2015). "Qualitative and descriptive research: Data type versus data analysis". In: *Language Teaching Research* 19.2, pp. 129–132. DOI: 10.1177/1362168815572747. eprint: https://doi.org/10.1177/1362168815572747. URL: https://doi.org/10.1177/1362168815572747.

Nelson, Laura K (2020). "Computational grounded theory: A methodological framework". In: *Sociological Methods & Research* 49.1, pp. 3–42.

Nguyen, Ha et al. (2021). "Establishing trustworthiness through algorithmic approaches to qualitative research". In: *International Conference on Quantitative Ethnography*. Springer, pp. 47–61.

Nielsen, Peter (2018). *Collaborative Coding of Qualitative Data(White Paper)*.

Noble, Helen and Joanna Smith (2015). "Issues of validity and reliability in qualitative research". In: *Evidence-Based Nursing* 18.2, pp. 34–35. ISSN: 1367-6539. DOI: 10.1136/eb-2015-102054. URL: https://ebn.bmj.com/content/18/2/34.

Norman, Geoff (2010). "Likert scales, levels of measurement and the "laws" of statistics". In: *Advances in health sciences education* 15.5, pp. 625–632.

O'Connor, Cliodhna and Helene Joffe (2020). "Intercoder reliability in qualitative research: debates and practical guidelines". In: *International journal of qualitative methods* 19, p. 1609406919899220. DOI: https://doi.org/10.1177/1609406919899220.

Olson, Gary M. and Judith S. Olson (Sept. 2000). "Distance Matters". In: *Hum.-Comput. Interact.* 15.2, pp. 139–178. ISSN: 0737-0024. DOI: 10.1207/S15327051HCI1523_4. URL: https://doi.org/10.1207/S15327051HCI1523_4.

OpenAI (2023). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL].

Oswald, Austin G (2019). "Improving outcomes with qualitative data analysis software: A reflective journey". In: *Qualitative Social Work* 18.3, pp. 436–442. DOI: https://doi.org/10.1177/1473325017744860.

Papenmeier, Andrea, Dagmar Kern, Gwenn Englebienne, et al. (2022). "It's Complicated: The Relationship between User Trust, Model Accuracy and Explanations in AI". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 29.4, pp. 1–33.

Papenmeier, Andrea, Dagmar Kern, Daniel Hienert, et al. (2022). "How Accurate Does It Feel?–Human Perception of Different Types of Classification Mistakes". In: *CHI Conference on Human Factors in Computing Systems*, pp. 1–13.

Paredes, Pablo et al. (2017). "Inquire: Large-scale early insight discovery for qualitative research". In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pp. 1562–1575.

Patel, Harshada, Michael Pettitt, and John R. Wilson (2012). "Factors of collaborative working: A framework for a collaboration model". In: *Applied Ergonomics* 43.1, pp. 1–26. ISSN: 0003-6870. DOI: https://doi.org/10.1016/j.apergo.2011.04.009. URL: https://www.sciencedirect.com/science/article/pii/S0003687011000573.

Pérez, I. J. et al. (2018). "On dynamic consensus processes in group decision making problems". In: *Information Sciences* 459, pp. 20–35. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.05.017. URL: https://www.sciencedirect.com/science/article/pii/S0020025518303724.

Porfirio, David et al. (2019). "Bodystorming human-robot interactions". In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pp. 479–491.

Qin, Li and Sue Kong (2015). "Perceived helpfulness, perceived trustworthiness, and their impact upon social commerce users' intention to seek shopping recommendations". In: *Journal of Internet Commerce* 14.4, pp. 492–508.

Rechkemmer, Amy and Ming Yin (2022). "When Confidence Meets Accuracy: Exploring the Effects of Multiple Performance Indicators on Trust in Machine Learning Models". In: *CHI Conference on Human Factors in Computing Systems*, pp. 1–14.

Richards, K Andrew R and Michael A Hemphill (2018). "A practical guide to collaborative qualitative data analysis". In: *Journal of Teaching in Physical education* 37.2, pp. 225–231. DOI: https://doi.org/10.1123/jtpe.2017-0084.

Rietz, Tim and Alexander Maedche (2021). "Cody: An AI-Based System to Semi-Automate Coding for Qualitative Research". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery. ISBN: 9781450380966. DOI: 10.1145/3411764.3445591. URL: https://doi.org/10.1145/3411764.3445591.

Saldaña, Johnny (2021). *The coding manual for qualitative researchers*. SAGE publications Ltd, pp. 1–440.

Scharowski, Nicolas et al. (2022). "Trust and Reliance in XAI–Distinguishing Between Attitudinal and Behavioral Measures". In: *arXiv preprint arXiv:2203.12318*.

Secure & Seamless Cloud Collaboration for Teams (n.d.). [EB/OL]. https://www.maxqda.com/teamcloud Accessed January 27, 2022.

Shneiderman, Ben (2020). “Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 10.4, pp. 1–31.

— (2022). *Human-centered AI*. Oxford University Press.

Smith, Jonathan A (2015). “Qualitative psychology: A practical guide to research methods”. In: *Qualitative psychology*, pp. 1–312.

Snyder, Hannah (2019). “Literature review as a research methodology: An overview and guidelines”. In: *Journal of Business Research* 104, pp. 333–339. ISSN: 0148-2963. DOI: https://doi.org/10.1016/j.jbusres.2019.07.039. URL: https://www.sciencedirect.com/science/article/pii/S0148296319304564.

Stumpf, Simone et al. (2009). “Interacting meaningfully with machine learning systems: Three experiments”. In: *International journal of human-computer studies* 67.8, pp. 639–662.

Sun, Yuqiang et al. (2023). “When GPT Meets Program Analysis: Towards Intelligent Detection of Smart Contract Logic Vulnerabilities in GPTScan”. In: *arXiv preprint arXiv:2308.03314*.

Tamm, Yan-Martin, Rinchin Damdinov, and Alexey Vasilev (2021). “Quality metrics in recommender systems: Do we calculate metrics consistently?” In: *Proceedings of the 15th ACM Conference on Recommender Systems*, pp. 708–713.

Teherani, A et al. (Dec. 2015). “Choosing a Qualitative Research Approach”. In: *J Grad Med Educ* 7.4, pp. 669–670. DOI: 10.4300/JGME-D-15-00414.1.

Thomas, Kenneth W (2008). “Thomas-kilmann conflict mode”. In: *TKI Profile and Interpretive Report* 1.11.

Vasconcelos, Helena et al. (2023). “Explanations can reduce overreliance on ai systems during decision-making”. In: *Proceedings of the ACM on Human-Computer Interaction* 7.CSCW1, pp. 1–38.

Vereschak, Oleksandra, Gilles Bailly, and Baptiste Caramiaux (2021). “How to evaluate trust in AI-assisted decision making? A survey of empirical methodologies”. In: *Proceedings of the ACM on Human-Computer Interaction* 5.CSCW2, pp. 1–39.

Vorm, Eric S (2018). “Assessing demand for transparency in intelligent systems using machine learning”. In: *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, pp. 1–7.

Wang, Dakuo et al. (2020). “From human-human collaboration to Human-AI collaboration: Designing AI systems that can work together with people”. In: *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pp. 1–6.

Watkins, Daphne C. (2017). “Rapid and Rigorous Qualitative Data Analysis: The “RADaR” Technique for Applied Research”. In: *International Journal of Qualitative Methods* 16.1, p. 1609406917712131. DOI: 10.1177/1609406917712131. eprint: https://doi.org/10.1177/1609406917712131. URL: https://doi.org/10.1177/1609406917712131.

Whang, Steven Euijong et al. (2021). "Data Collection and Quality Challenges in Deep Learning: A Data-Centric AI Perspective". In: *arXiv preprint arXiv:2112.06409*.

Wright, George and Peter Ayton (1988). "Decision time, subjective probability, and task difficulty". In: *Memory & Cognition* 16.2, pp. 176–185.

Wu, Tongshuang, Ellen Jiang, et al. (2022). "PromptChainer: Chaining Large Language Model Prompts through Visual Programming". In: *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI EA '22. New Orleans, LA, USA: Association for Computing Machinery. ISBN: 9781450391566. DOI: 10.1145/3491101.3519729. URL: https://doi.org/10.1145/3491101.3519729.

Wu, Tongshuang, Michael Terry, and Carrie Jun Cai (2022). "AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. <conf-loc>, <city>New Orleans</city>, <state>LA</state>, <country>USA</country>, </conf-loc>: Association for Computing Machinery. ISBN: 9781450391573. DOI: 10.1145/3491102.3517582. URL: https://doi.org/10.1145/3491102.3517582.

Xiao, Ziang (2023). "Seeing us through machines: designing and building conversational AI to understand humans". PhD thesis. University of Illinois at Urbana-Champaign.

Xiao, Ziang et al. (2023). "Supporting Qualitative Analysis with Large Language Models: Combining Codebook with GPT-3 for Deductive Coding". In: *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces*. IUI '23 Companion. Sydney, NSW, Australia: Association for Computing Machinery, pp. 75–78. ISBN: 9798400701078. DOI: 10.1145/3581754.3584136. URL: https://doi.org/10.1145/3581754.3584136.

Yan, Jasy Liew Suet, Nancy McCracken, and Kevin Crowston (2014). "Semi-automatic content analysis of qualitative data". In: *IConference 2014 Proceedings*.

Yin, Ming, Jennifer Wortman Vaughan, and Hanna Wallach (2019). "Understanding the effect of accuracy on trust in machine learning models". In: *Proceedings of the 2019 chi conference on human factors in computing systems*, pp. 1–12.

Zade, Himanshu et al. (2018). "Conceptualizing Disagreement in Qualitative Coding". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, pp. 1–11. ISBN: 9781450356206. DOI: 10.1145/3173574.3173733. URL: https://doi.org/10.1145/3173574.3173733.

Zhang, Chaoning et al. (2023). "A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need?" In: *arXiv preprint arXiv:2303.11717*.

Zhang, Zheng et al. (2023). *VISAR: A Human-AI Argumentative Writing Assistant with Visual Programming and Rapid Draft Prototyping*. arXiv: 2304.07810 [cs.HC].