Qualitative Code Suggestion: A Human-Centric Approach to Qualitative Coding

Cesare Spinoso-Di Piano

Computer Science
McGill University, Montreal

December 15, 2023

Acknowledgements

I would like to thank my supervisors, Jackie Cheung and Samira Rahimi, for their guidance and support throughout this project. Their patience and belief that I could see this project through to its completion was a tremendous source of motivation. I would also like to thank my lab mates for all the help and support they showed me over the last two years. I am particularly grateful to Jad Kabbara, who welcomed me so warmly into the group and to Malik Altakrori, who was always there to help me brainstorm ideas. I am also grateful to Andrei, Aylin, Caden, Ian, Jules, Martin, Sabina, Yu Lu and all the other members of our lab for having the patience to tolerate my sometimes eccentric nature.

Finally, it goes without saying that none of this could have been possible without my family. I would like to thank my mother and father, for always being there to listen and comfort me. I would also like to thank my uncle and my aunt for their support and encouragement and for always taking my calls. Lastly, I am forever indebted to my grandmother, who was my number one fan and supporter since I was little. I dedicate this thesis to her memory.

Abstract

Qualitative coding is a content analysis method in which researchers read through a text corpus and assign descriptive labels or *qualitative codes* to passages. To this day, it remains an arduous and manual process which human-computer interaction (HCI) studies have shown could greatly benefit from *assistive* methods powered by natural language processing (NLP) techniques. Yet, previous attempts at introducing language technologies into the qualitative coding process have framed it as a fully automatable task. In this thesis, we study a more human-centric approach to qualitative coding by defining the task of *qualitative code suggestion* (QCS) in which a list of previously assigned qualitative codes, ranked by relevance, is suggested for an identified passage. In addition to being user-motivated, QCS integrates previously ignored properties of qualitative coding such as the sequence in which passages are annotated, the importance of rare codes and the differences in annotation styles between coders. We investigate the QCS task by creating a qualitative coding dataset, `CVDQuoding`, consisting of interviews conducted with women at risk of cardiovascular disease and by modeling the QCS task with three paradigms: classification, information-retrieval and zero-shot prompting. In addition, we conduct a human evaluation to verify our experimental results and show that our systems are able to consistently make relevant code suggestions.

Résumé

Le codage qualitatif est une méthode d'analyse de contenu dans laquelle des chercheurs lisent un corpus de texte et attribuent des étiquettes descriptives, nommées *étiquettes qualitatives*, à des passages. À ce jour, cette méthode d'analyse demeure un processus difficile et manuel qui, selon des études faites dans le domaine d'intéraction humain-ordinateur, pourrait être facilitée par des techniques de traitement automatique de la langue naturelle (TALN). Or, les études qui ont déjà tenté d'introduire des technologies de traitement de langue dans le processus de codage qualitatif l'ont fait en considérant ce processus comme étant complètement automatisable. En revanche, dans cette thèse, nous étudions une approche au codage qualitatif centrée sur l'utilisateur en définissant la tâche de *suggestion de code qualitatif* (SCQ) dans laquelle une liste des codes qualitatifs déjà attribués par un codeur est ordonnée par pertinence et suggérée pour un nouveau passage identifié par le codeur. En plus d'être motivé par les désirs de codeurs qualitatifs, la tâche de SCQ intègre des propriétés du codage qualitatif ignorées jusqu'à maintenant par la communauté de codage automatique. Ces propriétés comprennent la séquence dans laquelle des passages ont été codés, l'importance des codes rares et la différence de style d'annotation entre codeurs. Nous étudions la tâche de SCQ en créant un ensemble de données de codage qualitatif que nous nommons `CVDQuoding` qui consiste d'entrevues avec des femmes à risque de maladies cardiovasculaires. Par ailleurs, nous modélisons cette tâche en utilisant trois paradigmes de la modélisation: la classification, la récupération d'information et la requête sans exemple. Finalement, nous menons une évaluation humaine pour vérifier nos résultats expérimentaux et nous démontrons que nos systèmes sont capables de suggérer des codes qualitatifs pertinents.

Contents

List of Figures

List of Tables

Contribution of Authors

Some of the results and the discussions found in Chapters 3, 4, 5 and 6 in this thesis appear in a publication at the 2023 conference on Empirical Methods in Natural Language Processing (EMNLP):

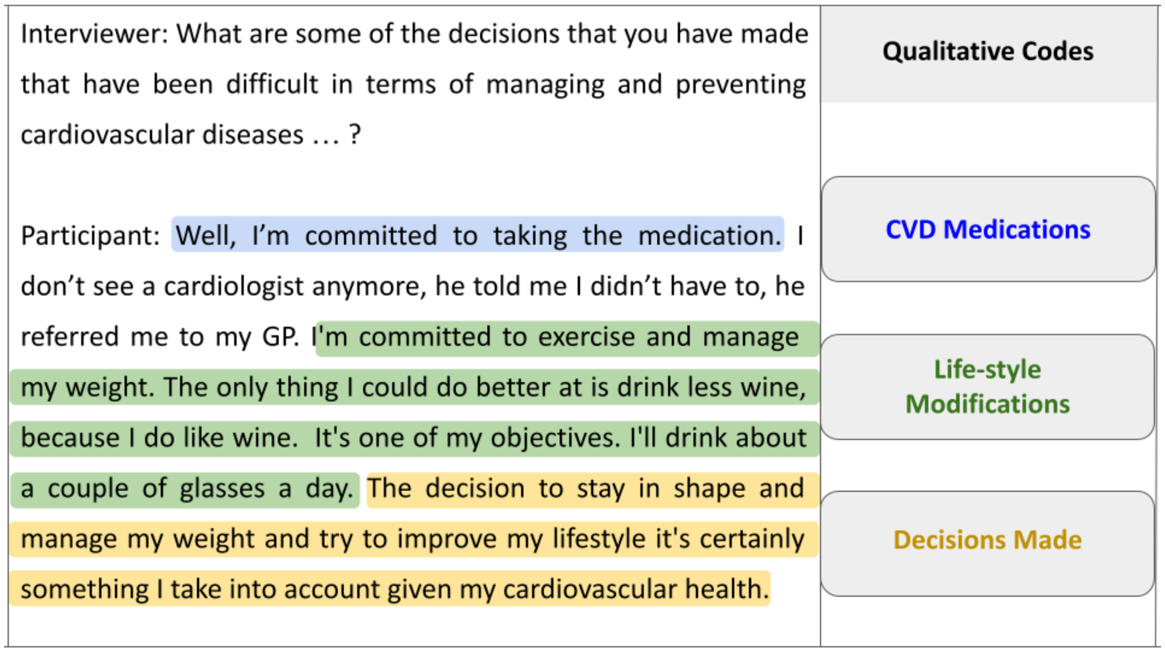
- Cesare Spinoso-Di Piano, Samira Abbasgholizadeh Rahimi, and Jackie Chi Kit Cheung. 2023. Qualitative Code Suggestion: A Human-Centric Approach to Qualitative Coding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14887-14909, Singapore. Association for Computational Linguistics.

1

Introduction

In qualitative research, qualitative coding is a content analysis method that is used to analyze textual corpora such as cultural and political texts, questionnaires and interview transcripts (Elliott, 2018). First used in social science studies, qualitative coding has become a universal method of analysis geared at providing researchers with an in-depth understanding of their studied corpus (Elliott, 2018). It is now employed in research disciplines such as education, psychology, and healthcare (Gough and Scott, 2000; Smith and McGannon, 2018; Chapman et al., 2015). During the coding process, a researcher carefully scans each document in their corpus, identifies passages that are associated with their research question and assigns these passages descriptive labels or *qualitative codes*.

For instance, in family medicine, researchers in Bousbiat et al. (2022) investigated the current state of cardiovascular disease (CVD) prevention and management in female patients in primary healthcare and the needs and required features of an AI technology for CVD prevention. To explore their research questions, the authors conducted interviews with 15 women at higher risk of cardiovascular disease (CVD). Once these interviews were conducted, their transcripts were analyzed via qualitative coding to answer the initial research question. In Figure 1.1, we show an excerpt from one of the interviews with a woman at risk of cardiovascular disease along with the annotations made by the researchers who conducted the qualitative coding. In this



Interviewer: What are some of the decisions that you have made that have been difficult in terms of managing and preventing cardiovascular diseases … ?

Participant: Well, I'm committed to taking the medication. I don't see a cardiologist anymore, he told me I didn't have to, he referred me to my GP. I'm committed to exercise and manage my weight. The only thing I could do better at is drink less wine, because I do like wine. It's one of my objectives. I'll drink about a couple of glasses a day. The decision to stay in shape and manage my weight and try to improve my lifestyle it's certainly something I take into account given my cardiovascular health.

Qualitative Codes

CVD Medications

Life-style Modifications

Decisions Made

Figure 1.1: Excerpt from an interview transcript from Bousbiat et al. (2022) along with the qualitative code annotations made by the qualitative researchers.

excerpt, the interviewer is asking the participant about the difficult decisions they have had to make to manage their risk of cardiovascular disease. The participant's answer contains several components which were broken down through the qualitative coding process. For example, their discussion of medications, "Well, I'm committed to taking the medication", is assigned the code "CVD Medications" while the discussion of "drink[ing] less wine" is assigned the code "Life-style Modifications". Ultimately, this method of analyzing their transcripts allowed the researchers of this project to make better recommendations on how to personalize CVD care for women.

Despite its widespread use and applicability, coding remains an arduous and time-consuming process as it requires researchers to manually annotate their studied corpus line by line. For instance, studies have shown that coding a 1-hour interview transcript will typically take an experienced researcher between 4 to 8 hours of annotation (Miles et al., 2019). In Bousbiat et al. (2022), it took 3 months for a single qualitative coder to complete the qualitative coding of the study's 15 1-hour interview transcripts.

As a result, the time-consuming nature of qualitative coding can limit the number of documents (e.g., interview transcripts) that are analyzed, which can in turn impact the strength of the conclusions made by studies employing qualitative coding (Anderson, 2010; Roberts et al., 2019).

As a result, to increase coding efficiency (i.e., the rate at which codes are assigned) and to reduce its cognitive load on researchers, practitioners from the natural language processing (NLP) community have attempted to introduce language technologies into the coding process. To do so, practitioners have primarily cast qualitative coding as a fully automatable text classification task in which codes are automatically assigned to passages without any additional human intervention (Crowston et al., 2012). However, as human-computer interaction (HCI) studies have shown (Rietz and Maedche, 2021), qualitative coders prefer receiving suggestions from assistive tools rather than having the entire corpus automatically coded for them. This aversion towards a fully automatic coding system may explain why, to this day, all available qualitative coding tools are used solely as "electronic filing cabinets" (Fielding and Lee, 2002). That is, as long as automatic coding is not approached and implemented in a user-oriented manner, coding tools with automated features such as NVivo[1] and MAXQDA[2] will continue to be used for their bookkeeping features only (Wiedemann, 2013; Marathe and Toyama, 2018).

1.1 Thesis Outline

In this thesis, we take a more human-centric approach to qualitative coding by identifying properties of qualitative coding ignored in previous automatic coding studies and by using these properties to build a user-oriented task which could assist qualitative coders' workflow. In particular, one of the key properties of qualitative coding

[1]https://lumivero.com/products/nvivo/
[2]https://www.maxqda.com/qualitative-data-analysis-software

includes the saturation of qualitative code assignments, the phenomenon by which most codes are discovered during the coding of the first few documents and are frequently re-used thereafter. In addition, we also study the importance of rare codes and the differences in qualitative coders' annotation styles. These three properties, which we show are important for a coding tool focused on assistance, are integrated into the *qualitative code suggestion* (QCS) task which we propose. The task of QCS requires a system to rank by relevance a list of previously assigned codes for a passage identified by a qualitative coder. The codes to rank only include those that have already been used by qualitative coders in previous documents; this is what is meant by *previously assigned* codes. In addition, since a passage may require an entirely new qualitative code, QCS contains an additional novel code detection subtask in which a passage must be labeled as 'novel' if it requires the creation of a new code or 'not novel' otherwise.

In addition to being better aligned with the nature of qualitative coding and with user preferences, QCS provides the NLP community with a new set of technical challenges which have remained unexplored in the field of automatic coding. For instance, our task definition exposes the necessity to evaluate code suggestions based on the sequence in which documents have been coded and the unsuitability of the commonly used i.i.d.[3] data distribution assumption. In addition, unlike other NLP tasks that use human-sourced annotations, QCS expects differences in annotation styles and avoids attempts to correct or homogenize these differences. Thus, although more challenging, we believe that this human-centric approach uncovers technical research avenues which are better positioned to positively impact researchers' qualitative coding workflow.

Along with proposing QCS, in this thesis, we create a dataset specifically tailored for this task, we explore the use and development of several NLP techniques to provide relevant code suggestions and we conduct additional human evaluations to verify

[3]independently and identically distributed

our findings. The dataset we create, which we name `CVDQuoding` (short for `CVD` `qualitative coding`), consists of the 15 transcripts of interviews with women at risk of CVD conducted in Bousbiat et al. (2022) along with the qualitative code annotations of 2 qualitative coders. Furthermore, we employ techniques from text classification, information-retrieval and large language model (LLM) prompting to create models which are able to make qualitative code suggestions as well as novel class detections. In addition, we conduct a human evaluation to verify the results of our models. The results of the human evaluation show that our modeling techniques are able to make relevant code suggestions and that there are several promising research directions in assisting qualitative coders with NLP in this human-centric manner.

1.2 Statement of Contributions

To summarize, in this thesis, we investigate the following research questions:

1. How can we formulate a human-centric task which better reflects the nature of qualitative coding and addresses the pain points exhibited by qualitative coders?
2. How effective are current NLP techniques at handling the challenges present in such a human-centric task, instantiated in this thesis as *qualitative code suggestion* (QCS)?

To this end, grounded in HCI studies on automated coding assistance, we propose the task of qualitative code suggestion (QCS) in which previously assigned codes are suggested for passages identified by the qualitative researcher. To investigate QCS, we create a qualitative coding dataset, named `CVDQuoding`, consisting of transcripts of interviews with women at risk of cardiovascular diseases along with the annotations of two qualitative coders. We experiment with classification, information-retrieval and zero-shot prompting techniques to model QCS and conduct a human evaluation which shows that our systems consistently make relevant code suggestions and that

QCS is a promising way of introducing NLP into the qualitative coding process in a human-centric manner.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 provides an overview of qualitative coding including the different types of coding and their use cases and describes previous work studying automatic qualitative coding techniques from complete automation to more user-oriented and assistive techniques. Chapter 3 details the properties of qualitative coding we focus on in this thesis and formalizes the qualitative code suggestion (QCS) task definition. Chapter 4 presents the `CVDQuoding` dataset which we create to study QCS and describes its main characteristics in connection with the qualitative coding properties we study in this thesis. Chapter 5 presents the data formatting and data modeling approaches we take to automate QCS. Chapter 6 presents the experimental setup we use to test our modeling approaches, the evaluation methods for our experiments and their results as well as an analysis of the results in connection with our research questions. Finally, Chapter 7 summarizes the findings of this thesis, presents some of its limitations and describes future work in this direction.

2 Background

In this chapter, we present an overview of qualitative coding in the context of content analysis methods as well as an analysis of the studies from the NLP and HCI communities which have aimed to introduce automation into the qualitative coding process. In particular, in Section 2.1, we present a formal definition of qualitative coding in the context of content analysis which is a family of research techniques centered around understanding the phenomena underlying text corpora. Furthermore, we describe the different characteristics of qualitative coding and its use cases in different fields. In addition, in Section 2.2, we summarize studies which have considered how automation can be introduced into the qualitative coding pipeline and what impact automation has on researchers' workflow. These studies range from presentations of methods which aim to fully automate the coding process, effectively taking away some - if not all - of the coding agency from the qualitative coder, to methods which aim to directly assist the qualitative coder as they are coding. Studies focusing on complete automation have mostly stemmed from the NLP community and have typically cast qualitative coding as a text classification task. In contrast, studies which have focused on assistive methods have mostly originated from the HCI community and have instead aimed to understand the pain points of qualitative coders and to develop coding tools with automated features which address those paint points.

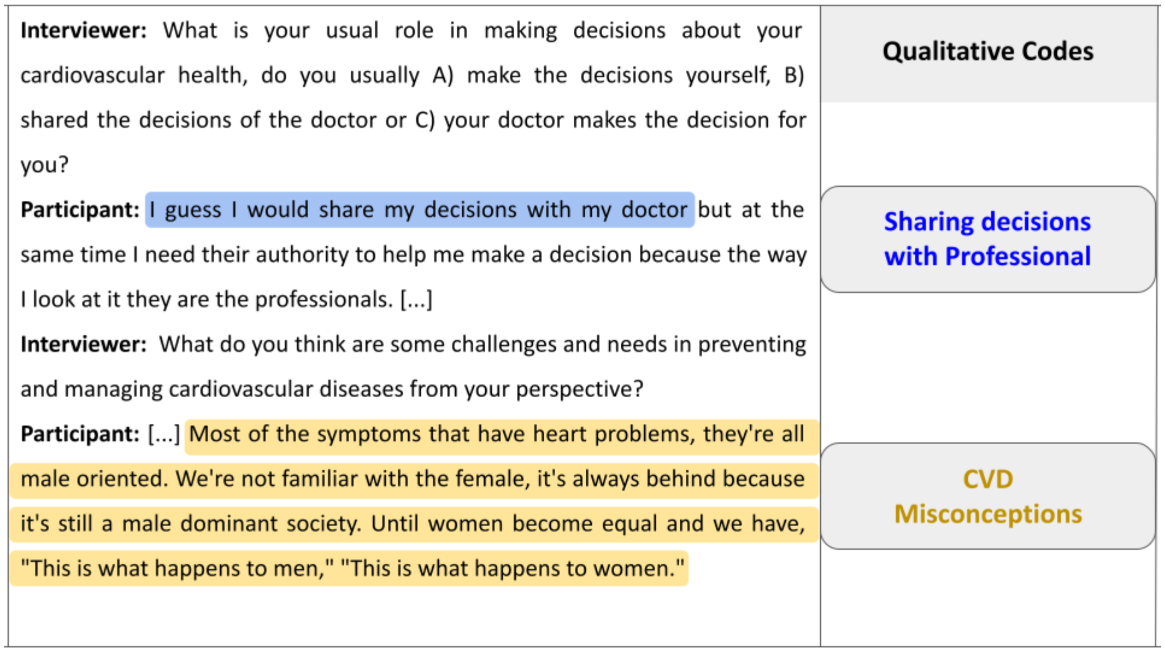
2.1 Qualitative Coding

In qualitative research, content analysis is defined as "a family of research techniques for making systematic, credible, or valid and replicable inferences from texts and other forms of communications" (Drisko and Maschi, 2016). These techniques range from simple word count and keyword visualizations to more thorough qualitative-coding-based studies. First used in the social sciences, qualitative coding has become a popular and commonly-used tool of analysis across various areas of research (Macnamara, 2006). Though the steps taken in conducting qualitative-coding-based content analyses can vary from one field to another, they all inevitably include two fundamental steps. The fist involves the qualitative researcher reading the studied content multiple times to get themselves familiarized with their data. The second involves extracting passages from the text that exhibit some idea relevant to the research question and assigning these passages descriptive labels or qualitative codes as illustrated in Figure 1.1 from Chapter 1. This latter step is often simply referred to as qualitative coding and is more precisely characterized in the following paragraphs.

Firstly, we define the notion of a qualitative code and the manner in which qualitative codes are created by researchers. As it stands, the exact definition that is given to a qualitative code may differ slightly from one qualitative researcher to another. This diversity in definitions is a result of different research fields each favoring slightly different variations of qualitative coding (Hsieh and Shannon, 2005). For our purposes, we will use the definition proposed by Saldaña (2021) who defines a **qualitative code** as "a word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute for a portion of language-based or visual data." Moreover, codes can either be formed **deductively** or **inductively**. In the deductive approach, the codes of the corpus are specified beforehand and are based off some hypotheses the researchers already have. Coding is then used as a means to validate (or invalidate) the stated hypotheses. In the inductive approach, the codes are not specified beforehand, but rather are generated directly from the data. Therefore, in the case of inductive coding, the creation of qualitative codes is more exploratory in nature.

Secondly, the creation of qualitative codes can be further distinguished based on the relation between the words used to form the code and the words in its correspondingly assigned text span(s). To qualify this relation, codes are generally described as either being **descriptive** or **in-vivo** (Gibbs, 2007). Descriptive codes provide a high-level description of what is being said in the highlighted passage and are closer to passage headlines or summaries (Gibbs, 2007). On the other hand, in-vivo codes typically repeat some part of the extracted text verbatim (Gibbs, 2007). A qualitative researcher may choose to use both descriptive and in-vivo codes throughout their annotation process. This diversity of code types allows for complimentary information to surface providing the researcher with a more well-rounded overview of their data. For example, in Figure 2.1, on one hand, the code "Sharing Decisions with Professionals" is more of an in-vivo code since it is an almost verbatim account of the highlighted passage. On the other hand, the code "CVD Misconceptions" can be classified as descriptive as it is much broader and tries to capture a higher-level idea than what is mentioned in the passage.

Finally, another dimension of qualitative codes pertains to how long the text spans associated with them tend to be and, thus, how much content is being captured by a single code assignment (Saldaña, 2014, 2021). As Saldaña (2014) describes, codes which cover more text will tend to *lump* similar or related low-level ideas together. As a result, codes of this nature will typically categorize larger sections of documents with broader descriptions. In contrast, codes which focus on shorter text spans such as sentences or phrases will *split* a document into much more detailed categories (Saldaña, 2014). Each approach has its advantages and disadvantages. **Lumping** tends to generate fewer codes which might expedite both the coding phase and the analysis phase. The downside of this approach is that the analysis might in turn



Interviewer: What is your usual role in making decisions about your cardiovascular health, do you usually A) make the decisions yourself, B) shared the decisions of the doctor or C) your doctor makes the decision for you?

Participant: I guess I would share my decisions with my doctor but at the same time I need their authority to help me make a decision because the way I look at it they are the professionals. [...]

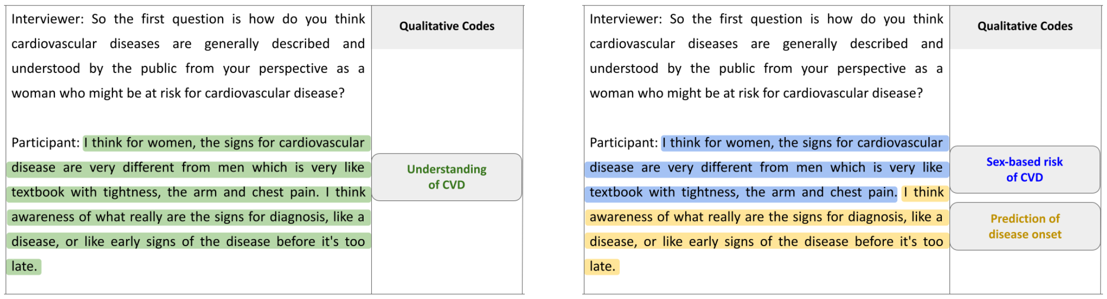
Interviewer: What do you think are some challenges and needs in preventing and managing cardiovascular diseases from your perspective?

Participant: [...] Most of the symptoms that have heart problems, they're all male oriented. We're not familiar with the female, it's always behind because it's still a male dominant society. Until women become equal and we have, "This is what happens to men," "This is what happens to women."

Figure 2.1: An excerpt from an interview transcript from Bousbiat et al. (2022) along with qualitative code annotations. The code "Sharing Decisions with Professionals" (top) is considered in-vivo code while the coder "CVD Misconceptions" (bottom) is considered descriptive.

overlook some important detail that could only be detected with a finer-grain coding scheme. **Splitting**, on the other hand, demands a much higher level of scrutiny and concentration from the researcher. However, this coding style decreases the chances that an important detail is overlooked once the researcher begins to analyze and aggregate their codes for insights. Therefore, a tradeoff exists between the cognitive load required to code text spans and the level of coding detail. For example, in Figure 2.2, we observe two distinct annotations for the same excerpt where the first one exhibits a lumping effect while the second exhibits a splitting effect.

After the qualitative researcher has finished coding their corpus, they will conduct several rounds of aggregation to distill their findings into more general **themes** connected to their initial research question (Braun and Clarke, 2006; Terry et al., 2017). The aggregation process occurs by examining the codes and the passages to which they were assigned and grouping codes that exhibit commonalities, typically based on



Interviewer: So the first question is how do you think cardiovascular diseases are generally described and understood by the public from your perspective as a woman who might be at risk for cardiovascular disease?

Participant: I think for women, the signs for cardiovascular disease are very different from men which is very like textbook with tightness, the arm and chest pain. I think awareness of what really are the signs for diagnosis, like a disease, or like early signs of the disease before it's too late.

Interviewer: So the first question is how do you think cardiovascular diseases are generally described and understood by the public from your perspective as a woman who might be at risk for cardiovascular disease?

Participant: I think for women, the signs for cardiovascular disease are very different from men which is very like textbook with tightness, the arm and chest pain. I think awareness of what really are the signs for diagnosis, like a disease, or like early signs of the disease before it's too late.

Figure 2.2: An excerpt from an interview transcript from Bousbiat et al. (2022) along with qualitative code annotations of two coders. The annotation on the left exhibits a "lumping" characteristic while the one on the right exhibits a "splitting" characteristic.

what research question is being studied. These groups of codes will each be assigned higher-level descriptions and this process will be repeated until the researchers believe that they have formed a handful of well delineated themes. The primary difference between a code and a theme is their relation with the studied corpus. While a code will always be connected to some idea explicitly evoked in the studied document, a theme should describe an underlying pattern evidenced by the studied corpus. That is, a theme should not simply summarize some portion of the corpus. Rather, it should make an analytic claim about the corpus using the codes and text spans as evidence. For example, a code could be "Level of interest in tracking health information" whereas a theme related to this code could be "Any type of information tracking raises privacy concerns". The themes that result from the researcher's qualitative analysis will be used to argue a certain position with respect to their research question(s). A visualization of this process can be observed in Figure 2.3 in which codes are progressively aggregated into themes.

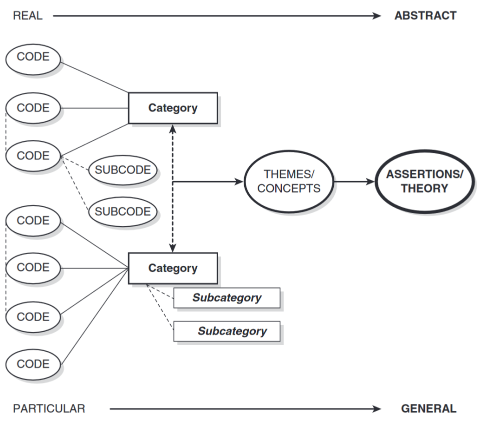


Figure 2.3: An illustration of the process by which codes are aggregated into themes (Saldaña, 2021).

Since its first uses in sociology and journalism (Drisko and Maschi, 2016), content analysis in the form of qualitative coding has become a standard method of analysis in multiple domains including education, psychology, and medicine. In education, Górska et al. (2018) used qualitative coding to analyze undergraduate students' thank you letters to their teachers to understand the significance and the impact of teacher-student interactions. The authors of the study discovered that undergraduates value teachers with whom they can interact with on topics such as homework and test feedback as well as career advice. In psychology, Grantham et al. (2015) aimed to examine when and why therapists working with bilingual Spanish-English-speaking Latino clients switched from speaking English to Spanish, and vice-versa. To do so, 9 Spanish-English speaking therapists with bilingual Latino clients were interviewed in a semi-structured manner after which the interviews were transcribed and qualitatively coded. The study concluded that therapists switch languages to establish a sense of trust with their clients. In medicine, Santiago-Rivera et al. (2009) conducted a meta-analysis of studies that used qualitative coding to analyze testimonials (e.g., transcribed semi-structured interviews) from people with dementia (PWD). The meta-analysis showed that, in many cases, the qualitative coding of PWD testimonials revealed that people strive to maintain continuity in their lives despite their diagnosis by engaging in social activities and by contributing to their households and their communities.

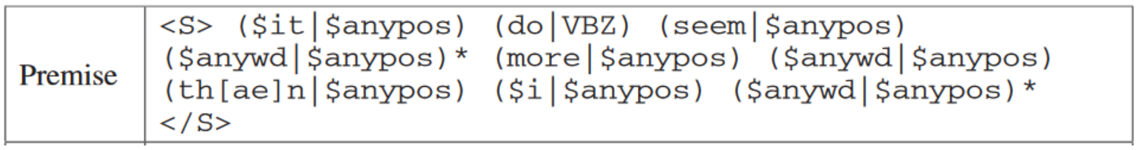
2.2 Qualitative Coding and Automation

Despite its ubiquity, content analysis using qualitative coding has remained a time-consuming and expensive endeavour limiting both the breadth and the depth of studies employing this analysis technique (Patton, 2014; Bengtsson, 2016). As a result, there has been a concerted effort from both the NLP and HCI community to develop automated assistive coding techniques and study the impacts of these techniques on qualitative coders' workflow. In Section 2.2.1, we describe techniques which have been proposed by NLP researchers to automate most, if not all, of the coding pipeline. In Section 2.2.2, we describe work from the HCI community which leverage the interaction between the qualitative coders and the automated features to assist them in their coding of corpora.

2.2.1 The Fully Automatable Perspective

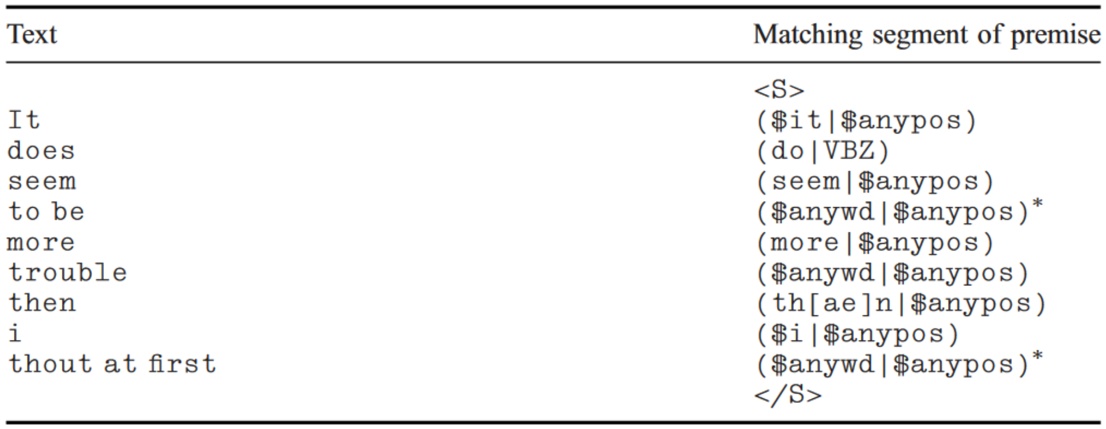
Thus far, the NLP community has treated the process of qualitative coding as a fully automatable task using traditional, as well as more advanced, text classification techniques to assign pre-defined codes to text spans. We first discuss the most notable and prominent work related to this perspective of automation and qualitative coding and then briefly discuss why a lack of coder intervention is not necessarily a desirable property in a coding system.

The first attempts at introducing NLP techniques into the qualitative coding process casted *deductive* qualitative coding as a text classification task and used rule-based approaches to model this task. In Crowston et al. (2010, 2012), the authors explored the effectiveness of rule-based NLP techniques to assist qualitative researchers in the deductive coding of messages between developers working on open source software projects. In total, 3011 segments of text were assigned codes from a list of 15 pre-specified codes. The text segments were then split into a training and testing set and an NLP researcher was tasked with developing rules to match each text segment with its correct code using features such as part-of-speech tags and keywords. In Figures 2.4 and 2.5, we show an example of a hand-crafted rule for the "Agreement" code and how it was used to match this code with a text segment. The authors measured their system's performance using recall and precision scores and showed that it did well for codes such as the "Formality" and "Appreciation" codes. However, their system struggled on most of the other codes and the manual creation of classification rules was a slow and unscalable process. As a result, in the next paragraph, we present follow-up work which aimed to move away from rule-based text classification approaches and towards supervised-learning-based approaches.



Premise	`<S> ($it\|$anypos) (do|VBZ) (seem|$anypos) ($anywd|$anypos)* (more\|$anypos) ($anywd\|$anypos) (th[ae]n|$anypos) ($i|$anypos) ($anywd|$anypos)* </S>`

Figure 2.4: An example of one of the hand-crafted rules to match codes with text segments from Crowston et al. (2012). This rule is for the "Agreement" code and uses part-of-speech tags such as VBZ to indicate a verb (in the 3$^{\text{rd}}$ person singular form) as well as variables to represent semantic classes of words such as `$it` which is meant to capture function words such as 'it' and 'this'.



Text	Matching segment of premise
	<S>
It	($it\|$anypos)
does	(do|VBZ)
seem	(seem|$anypos)
to be	($anywd\|$anypos)*
more	(more|$anypos)
trouble	($anywd\|$anypos)
then	(th[ae]n|$anypos)
i	($i\|$anypos)
thout at first	($anywd\|$anypos)*
	</S>

Figure 2.5: An example of a matching between a coded text segment from the test set and the rule for the code "Agreement" (described in Figure 2.4) from Crowston et al. (2012).

Follow-up studies to Crowston et al. (2010, 2012) continued to cast deductive qualitative coding as a text classification task but used supervised learning to predict the correct code assignment for a given text span. In McCracken et al. (2014); Liew et al. (2014), the authors shifted from crafting rules for each code to training a support vector machine (SVM) to predict the correct code assignments of text spans. In particular, the authors used the same dataset as Crowston et al. (2012) and framed the assignment of codes to text spans as a multi-label binary classification task. The multi-label setting was needed because certain text spans were assigned multiple codes. The authors trained a different SVM for each pre-specified code using features like the unigram and bigram counts of text spans as well as the part-of-speech tags present in the text spans. Using precision and recall scores, the authors showed that supervised learning was more flexible than rule-based learning. However, the former method struggled to match the performance of the latter method for codes with a small number of training instances. In Scharkow (2013), the authors used supervised text classification to automatically code German news articles and made the same observations as McCracken et al. (2014) for Bayesian classifiers. Lastly, in Kaufmann et al. (2020), the authors explored the use of semantic similarity scores between codes and text spans computed using keyword overlap to automatically assign codes to passages.

More recent work in automatic qualitative coding has investigated the use of neural networks and large language models (LLMs) such as BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020) to automatically assign codes to text spans. In Grandeit et al. (2020), the authors compared the performance of an SVM classifier with neural-based classifiers such as BERT and DistilBERT which have been shown to perform significantly better than SVMs on several text classification tasks (Kowsari et al., 2019). For their study, social scientists qualitatively coded posts made on a German parent counseling website using a pre-defined codebook of 50 codes. The coding of these posts resulted in 10000 text spans annotated with one or more codes. Using F_1 scores to compare performance, the authors showed that a BERT classifier tends to do better than an SVM classifier when classifying codes with a large number of associated text spans. Furthermore, they showed that even if the labels predicted by the BERT classifiers do not match the initial coder's annotations the predictions made by the models may still be relevant and, therefore, useful. In Xiao et al. (2023), the authors used a prompt-based learning approach (Liu et al., 2023) to predict the code assigned to a text span. More specifically, the authors created a dataset of curiosity-driven questions asked by children which were coded by a team of psychologists. The NLP researchers used the codebook defined by the psychologists as well as text spans associated with each code in the codebook as a prompt for the GPT-3 model. The researchers then used GPT-3 to generate the code predictions for each text span in their test set. By using Cohen's Kappa, the authors showed that GPT-3 was able to achieve fair to substantial agreement with the psychologists' coding decisions.

To summarize, the NLP community has geared most of its efforts towards fully automatic (deductive) coding systems which, given some corpus to code, assign pre-defined codes to spans of text with the help of a text classifier. Effectively, this removes the researcher from the core coding process and shifts their responsibility to verifying the correctness of code assignments. This shift in the duties and responsibilities of the coder is what characterizes this *fully automatable* perspective. One notable exception to this view is work by Bakharia (2014); Bakharia et al. (2016) who build an *interactive* topic model which allows qualitative coders to split and merge clusters of text spans and associate them with different codes. In this case, the qualitative coders preserved most of the coding agency and, as a result, created clusters of text spans which were better aligned with their research questions than the clusters created using other non-interactive methods. In fact, as we will discuss in the upcoming section, coders prefer coding most of their corpus themselves and only desire automation under certain conditions (e.g., the codes have become repetitive).

2.2.2 The Assistive Perspective

There has been extensive work, mostly from the HCI community, in understanding how qualitative coders would like automation and artificial intelligence (AI) to *assist* them during their coding process (Grimmer and Stewart, 2013; Muller et al., 2016; Chen et al., 2018; Marathe and Toyama, 2018; Jiang et al., 2021a; Feuston and Brubaker, 2021). In particular, several studies have shown that while qualitative researchers desire automated assistance, they insist that any kind of assistance must not take away their coding and analysis agency (Marathe and Toyama, 2018; Jiang et al., 2021a; Feuston and Brubaker, 2021). That is, regardless of the sophistication of the qualitative coding tool, researchers must remain the primary coders of their corpus. This agency requirement can be explained by the fact that researchers do not simply use qualitative coding as a means to create an aggregated view of their corpus. Rather, researchers value the process of qualitative coding itself because it cultivates their understanding of the phenomena underlying their corpus (Marathe and Toyama, 2018; Jiang et al., 2021a; Feuston and Brubaker, 2021). For instance, Jiang et al. (2021a) showed that the process of creating and assigning concise and reusable codes to passages forces researchers to constantly doubt their comprehension of their corpus' most salient themes. The authors showed that this cultivation and resolution of self-doubts is one of the objectives of qualitative coding as it often improves the quality and conciseness of codes and, subsequently, themes. In light of these findings, Marathe and Toyama (2018) showed that researchers desire assistance but only once the researchers have become familiar with their dataset and have annotated large portions of it.

As a result, several papers have taken a more assistive perspective to automatic coding by favoring *coder-in-the-loop* systems which make suggestions, instead of decisions, based on a coder's previous code assignments and which allow coders to interact with the system to improve suggestions (Chen et al., 2016, 2018). For example, in Drouhard et al. (2017), the authors moved away from making code assignment predictions and, instead, moved towards suggesting ambiguous code assignments. In their study, a code assignment was considered ambiguous either due to the linguistic ambiguity of a text span or the subjective nature of coding (e.g., two coders with different backgrounds understanding a text span differently). By using the disagreement between coders as a proxy for ambiguity, the authors were able to create a system that could learn from the disagreements of annotators and suggest ambiguous passages that should be analyzed and discussed in more detail. Overall, this shift towards detecting ambiguous code assignments broadened researchers' understanding of their corpus and allowed the qualitative coding tool to better "preserv[e] the human-centered nature of qualitative analysis" (Drouhard et al., 2017). In another human-AI collaborative system, Rietz and Maedche (2021) developed an interactive machine learning (IML) system which used both user-defined rules as well as supervised learning (SL) to retrieve text spans related to a given code. Users could then accept or reject suggested text spans which would further train the SL model and could also modify the rule definition to refine the precision and recall of the retriever. Although their experiments did not show improvements in coder efficiency, the qualitative coders that tested the system reported that the quality of their codes improved as the construction of rules forced them to carefully think about the meaning of their codes.

In this thesis, we aim to take a similar assistive user-oriented approach to automatic qualitative coding by defining a new task which incorporates several properties of qualitative coding and which addresses several common coding pain points. We discuss this further in Chapter 3.

3

The Qualitative Code Suggestion Task

In this chapter, we discuss the properties of qualitative coding which we use to define the *qualitative code suggestion* (QCS) task. Specifically, in Section 3.1, we discuss the saturation of qualitative codes, the existence and importance of rare codes and the difference in qualitative coder annotation styles as being important properties to consider for the task we define. In Section 3.2, we discuss the formalization of the QCS task which consists of the main code ranking task in which previously assigned codes are ranked, in order of relevance, for a highlighted passage and the novel code detection subtask in which a highlighted passage is categorized as either requiring or not requiring a novel code.

3.1 Properties of Qualitative Coding

In qualitative coding, researchers code their corpus to gain a deeper understanding of a certain phenomenon related to their research question. As a result, in deductive and inductive coding, the first few documents that are coded are critical for the researcher's *own* understanding. Once the researchers have read through enough documents and have become familiar with their corpus, well-documented properties of qualitative coding arise (van Rijnsoever, 2017; Saunders et al., 2018; Loslever et al., 2019). Furthermore, as studies have illustrated (Marathe and Toyama, 2018; Rietz and Maedche, 2021), these properties should be considered when developing assistive coding techniques. Thus, in the following paragraphs, we describe properties of qualitative coding which have been under-explored in automatic qualitative coding methods and which we integrate in the task definition of this thesis.

Data saturation. As noted in previous studies (Marathe and Toyama, 2018; Rietz and Maedche, 2021), coders that carry out qualitative coding reach a personal point of *data saturation* after which few new codes will appear (Saunders et al., 2018). This well-known property has been documented as a major pain point for qualitative coders. As Figure 3.1 highlights, once qualitative coders reach their personal point of data saturation they are no longer coding to deepen their understanding of their corpus, but rather to gain more evidence for the observations they made during the initial, more exploratory, coding phase (Marathe and Toyama, 2018).

> Once participants gained a sense of what the data contained by going through the process of codebook development, they found it tedious to code the rest, as Rachel explains: “When you reach saturation, in those last interviews, you’re like, ‘We know. We know. We got all this, cool, in the bag.’ But you still have to code them. And there are so few interesting nuggets... for the most part you’re seeing the same things.”

Figure 3.1: Excerpt from Marathe and Toyama (2018) which presents a user’s perspective on data saturation.

As studies have shown (Marathe and Toyama, 2018; Rietz and Maedche, 2021; Jiang et al., 2021a; Feuston and Brubaker, 2021), qualitative coders desire automation *past* this point of data saturation. For instance, coders desire systems which automatically suggest several relevant codes for passages that they highlight as they code. These suggestions could help the qualitative coder consider relevant code assignments without needing to sift through irrelevant codes. At the same time, suggesting several relevant codes reduces the shift towards complete automation which, as discussed in Section 2.2.2, qualitative researchers do not desire.

Rare codes. Even though new codes may be rare past the point of data saturation, their occurrences remain possible and scientifically important to the coders (van Rijnsoever, 2017). That is, the rarity of a code is in and of itself an important scientific finding. Thus, while suggesting relevant codes may alleviate some of coding's cognitive load, detecting *rare codes* may be just as important. This property has remained relatively unexplored in automatic coding with most technical approaches framing rare codes as codes with few supporting examples and, therefore, "difficult" to classify correctly (Crowston et al., 2012; Grandeit et al., 2020).

Annotation style. Since coding is meant to cultivate a particular researcher's understanding of a corpus, no two coders will have the same *annotation style* (Loslever et al., 2019). For instance, as discussed in Section 2.1, two qualitative coders may have different ways of assigning codes to passages with one favoring a splitting style and another favoring a lumping style. Thus, previous efforts to homogenize code annotations, for instance by grouping similar codes from different coders together, should be avoided. Although convenient for modeling purposes, this merging is not supported by the user-oriented approach we take in this thesis.

3.2 Task Definition

Now that we've established the properties of qualitative coding which we believe are fundamental in creating a user-centered task related to automatic coding, we discuss the actual QCS task that we study and model in this thesis. The QCS task is composed of the main code ranking task as well as the novel code detection subtask and is suitable for both inductive and deductive qualitative coding.

Code ranking. The task of QCS mainly involves ranking previously assigned codes for passages highlighted by a qualitative researcher in order of relevance. Formally, consider the setting in which a corpus is comprised of N documents $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$. Each document, d_i, is a set of text spans $\{t_n\}_{n=1}^{\texttt{len}(d_i)}$ which have been identified and assigned codes by a qualitative coder. Now, suppose $K \in \{1 \ldots N\}$ is a *particular* coder's point of *data saturation*. In this case, we collect all the codes assigned to text spans from documents d_1 to d_K and create a set of *previously assigned* codes $\mathcal{C} = \{c_1, \ldots, c_m\}$. The main code ranking task that we define here consists of ranking, in order of relevance, the set of codes in $\mathcal{C}$ with respect to every identified text span in the set of documents past the point of data saturation, i.e. $\{t \in d_i : i = K + 1 \ldots N\}$.

Novel code detection. In the novel code detection subtask, each passage highlighted by a qualitative coder must be categorized as either 'novel' or 'not novel'. Formally, a passage should be categorized as 'novel' if and only if one of its truly assigned codes[1] does not belong to $\mathcal{C}$. This subtask is necessary because QCS assumes a sequential coding of the documents from 1 to N and, therefore, some rarer codes may only surface in the test set portion of the corpus $\{d_{K+1}, \ldots, d_N\}$. The addition of this subtask addresses the property discussed in Section 3.1 that discovering rare codes may be just as important as suggesting relevant ones.

Given that qualitative coders do not themselves annotate passages as 'novel' or 'not novel', we must design a heuristic which creates training and testing instances for this subtask. To create training instances, QCS prescribes assigning the novel code to all text spans in the training set that have been assigned a code with a training frequency of 1. All of the text spans in the validation and test sets with at least one assigned code that does not appear in the training set are assigned the novel code. This implies that the number of novel code instances directly depends on the coder's annotation style e.g., if a coder rarely creates new codes then there may be very few 'novel' code instances.

[1] Recall that several codes may be assigned to the same text span.

4

The CVDQuoding Dataset

In this chapter, we describe the CVD qualitative coding dataset which we name CVDQuoding and which we use to investigate the QCS task. We describe how it was created and draw parallels between CVDQuoding and the properties of qualitative coding we described in Section 3.1.

4.1 Dataset Creation

CVDQuoding is a dataset consisting of 15 transcripts of interviews conducted with women at risk of cardiovascular diseases (CVDs) as well as annotations carried out by two qualitative coders during their *inductive* coding of the corpus of transcripts. These qualitative coding annotations consist of text spans identified by the qualitative coders as well as the codes that were assigned to each text span. Since codes assigned to a text span may change as the researchers review their work, we chose to extract the final code assigned to each text span. Additional details about the dataset as well as comparisons with previous qualitative coding datasets, which have all been closed-source and thus could not be used for this thesis, can be found in Table 4.1. We observe that previous studies have had access to a larger number annotations than are available in CVDQuoding. This difference in dataset size might be attributed to the fact that previous work has focused on deductive qualitative coding when

Source	Corpus size (in number of words)	Number of annotations	Publicly available?
(Crowston et al., 2012)	84870	3011	No
(Grandeit et al., 2020)	N/A	10000	No
(Xiao et al., 2023)	N/A	668	No
`CVDQuoding`	63927	1175	Yes

Table 4.1: Comparative analysis of our dataset with previously closed-source datasets. In this case, the number of words are counted using space separation and the number of annotations refer to the number of text spans that have been assigned one or more codes.

applied to many short documents such as instant messages and forum posts (Crowston et al., 2012; Grandeit et al., 2020). However, in our case, there are only a few coded documents with relatively longer lengths averaging 4262 words per transcript.

The annotations in `CVDQuoding` were originally created as part of a larger project[1] to investigate the needs of women Bousbiat et al. (2022) and primary care providers Sandhu et al. (2023) in designing AI tools to manage and prevent CVDs and were subsequently adapted for the QCS task. In the original work, 15 women were interviewed and asked 20 scripted questions. These questions were related to the health challenges they faced as well as their opinions about using AI tools to help them manage their CVD risk. In Table 4.2, we provide a sample of the list of questions asked throughout the interviews. Some of the questions were open-ended such as questions 1 and 5 while other questions like question 10 were more narrow and simply asked for the interviewee to state their level of interest in a certain feature (For the full list of questions see Table A.1 in Appendix A).

For the purposes of this thesis, we extracted the transcripts corresponding to the 15 interviews and tagged the text spans coded by each coder and the questions asked by the interviewer using XML. In Figure 4.1, we present an excerpt from one

[1]https://rahimislab.ca/

Question 1	How do you think cardiovascular diseases are generally described and understood by the public?
Question 5	What do you think are some challenges and needs in preventing and managing cardiovascular diseases (from your perspective as a woman at risk of CVD)?
Question 10	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in a step-count feature?
Question 15	Would you like to be able to follow your progress and receive push-notifications through the Xi-Care tool?
Question 20	Is there something else you'd like to add about ethical aspects in regards to the Xi-Care tool that will be empowered by AI (Justice; Non-maleficence; Autonomy; Beneficence; Explicability/Transparency)?

Table 4.2: Sample list of questions asked during the interviews of the `CVDQuoding` dataset. *Xi-Care* is the name of the app being proposed to participants of the study to help them control their risks of cardiovascular diseases (CVDs). The full list of questions can be found in Table A.1.

```
<question_2>Interviewer: Have you ever been informed about your
   cardiovascular health, and if so how have you been informed?</
   question_2>

Participant 1: <code coder="2" value="Understanding of CVD">So, I've
   been in cardiology since '96. I used to work at the Montreal Heart
   Institute before so <code coder="1" value="Sources of information">
   I attended cardiac rounds and went to conferences and things like
   that. I also like to read on health, so media, papers and then
   conferences.</code></code>
```

Figure 4.1: Excerpt from Interview 1 of the `CVDQuoding` dataset with XML tags to identify the question asked and the text spans that have been annotated. In the `code` tag, the attribute `coder` identifies which qualitative coder was responsible for the annotation and the attribute `value` identifies the code assigned to the text span.

of the interview transcripts along with the additional XML tags that we created to represent the annotations of both qualitative coders. The excerpt shows how we tagged the question asked by the interviewer and the annotations made by both qualitative coders.

4.2 Dataset Characteristics

`CVDQuoding` exhibits characteristics that are reflective of the properties of qualitative coding presented in Section 3.1. In terms of *data saturation*, both coders create many new codes in the first five interviews and this trend tapers off as they code more transcripts (Figure 4.2). In addition, as can be observed from Figure 4.2, coder 1 finds new *rare codes* in every transcript whereas coder 2 ceases to create new codes by interview 7. Moreover, both coders clearly exhibit significantly different *annotation styles* with coder 1 creating 207 codes with an average of 4 text spans per code and coder 2 creating 23 codes with an average of 19 text spans per code. This difference can be attributed to the specificity of coder-2's codes, which tend to be more abstract (e.g., "Health Measures Taken") than coder-1's codes (e.g., "CVD-related examinations, tests or measures"). For additional details about `CVDQuoding`, including a sample list of codes, see Section A.2 in Appendix A.

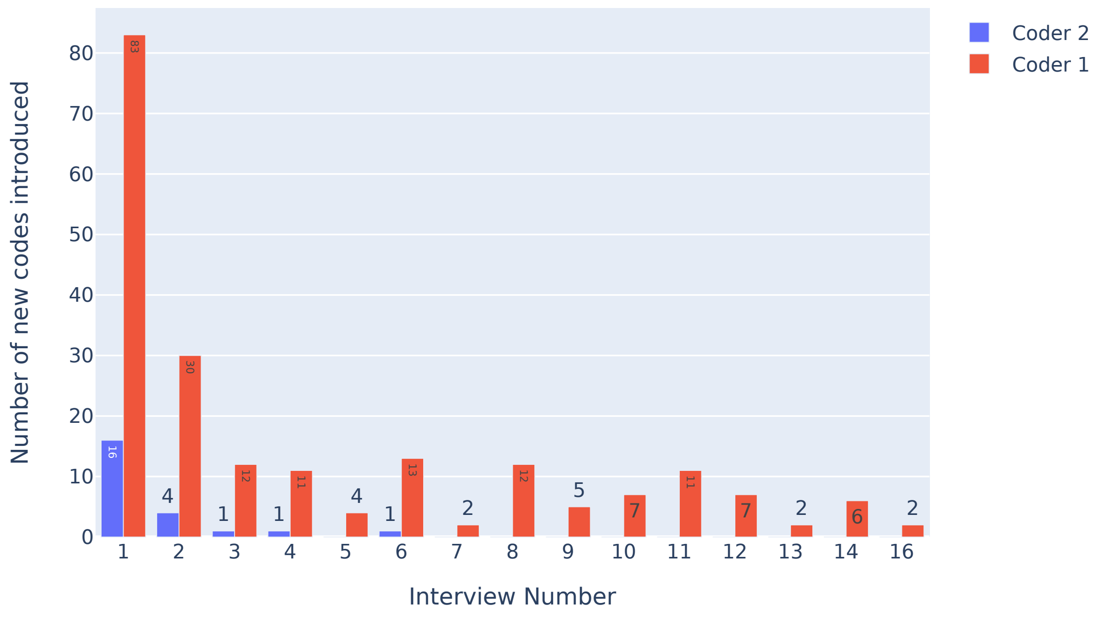


Figure 4.2: Distribution of the number of new codes introduced per interview for each coder.

5

Modeling QCS

In this chapter, we describe the different techniques we use to model the QCS task which, recall, refers to both the main code ranking task as well as the code detection subtask. To this end, in Section 5.1, we describe our data formatting decisions and, in Section 5.2, we formally present the modeling paradigms we explore which include classification, information-retrieval and zero-shot prompting.

5.1 Data Formatting

For all three modeling paradigms, we associate every contiguous text span $t \in d_i$ that was coded by one of the qualitative coders with its immediate context c. In the case of `CVDQuoding`, we define the context c as being the question q asked right before t. Thus, in our case, every model has access to a tuple (q, t) consisting of a question, q, and a text span, t, highlighted by a qualitative coder. We use the previous question as an approximation of the context as most coding decisions in `CVDQuoding` involve considering a text span as the answer to a preset question. For example, the code "Interest in AI tools" is assigned to "Very interested, yes" because it is the answer to "Are you interested in using AI tools for CVD?". The decision to augment text spans with their neighboring context is supported by previous work (Marathe and Toyama, 2018) which found that coders' assignment of codes to text spans is highly context dependent and that modeling this context is necessary to achieve acceptable levels of precision and recall.

5.2 Modeling Paradigms

To model the QCS task, the methods we develop must both rank lists of codes by relevance and detect novel codes from text spans. To this end, we choose to explore three distinct modeling paradigms drawn from the NLP literature: classification, information-retrieval and zero-shot prompting. The classification paradigm, which is inspired by previous work (Crowston et al., 2012; Grandeit et al., 2020; Xiao et al., 2023), assumes that the confidence scores of a multi-label classifier can be used to both rank codes and classify novel code instances. The information-retrieval paradigm assumes that codes can be treated as documents and that text spans are the queries used to rank them. Finally, the zero-shot prompting paradigm assumes that the descriptiveness of codes is sufficient for a generative LLM such as GPT-3 (Brown et al., 2020) to rank them based on the text span. The advantage of the latter two approaches over the traditional classification approach is they can leverage the additional information found in the description of the codes to improve their relevance-based ranking.

5.2.1 Classification Paradigm

We build a $|\mathcal{C}| + 1$ multi-label binary classifier trained on the first K annotated transcripts to predict the binary assignment of each code, including the novel code, to a test text span. We use the scores produced for each code to sort the set of codes $\mathcal{C}$ for a test instance $(q, t) \in \{(q, t) : t \in d_i, i = K + 1 \dots N\}$ to create a ranked list of codes. Furthermore, an instance is assigned the novel code only if it is given the highest classification score among all codes.

5.2.2 Information-retrieval Paradigm

In the information-retrieval paradigm, we treat an instance (q, t) as a query and use two neural-retrieval architectures to rank the set of codes $\mathcal{C}$ with an additional step for the novel code detection subtask.

For the main ranking task, we build two neural-retrieval architectures originally presented by Reimers and Gurevych (2019): the bi-encoder and the cross-encoder. In the bi-encoder, a representation is learned for the text span t, the previous question q and the code c. The representations of the text span, h_t, and the previous question, h_q, are max pooled together and a score is computed by applying a cosine similarity between the pooled representation and the code representation, h_c. In the cross-encoder, representations are learned for the concatenations of the code with the question, q [SEP] c[1], and with the text span, t [SEP] c. The representations, $h_{q \text{ [SEP] } c}$ and $h_{t \text{ [SEP] } c}$ are max pooled together and a classification head is placed on top of the pooled representation to produce scores. In both cases, the code scores produced are used to rank all but the novel code.

For the novel code detection subtask, a classification head is trained on top of the vector of scores $\hat{y}_i$ which consists of the scores computed between each code $c \in \mathcal{C}$ and the input instance $(q, t)_i$. Thus, if there are 10 codes in the training set (excluding novel), then for every instance $(q, t)_i$ the classifier is passed a 10-dimensional vector $\hat{y}_i$ computed from either the bi-encoder or cross-encoder.

5.2.3 Zero-shot Prompting Paradigm

In the zero-shot prompting paradigm, we provide an autoregressive LLM M with a prompt containing general instructions, the list of codes $\mathcal{C}$ and a text span t to code along with its previously asked question q. In Figure 5.1, we show the template that we use to prompt the LLM M. Upon generation, the suggested codes are extracted

[1] $[SEP]$ is a special separation token

> You are a helpful assistant that suggests qualitative codes for a qualitative researcher. The coders you can suggest are: [AVAILABLE_CODES], and None of the above. Which of the previous codes would you assign to the following excerpt from an interview with a woman at risk of cardiovascular disease (CVD): "Question: [QUESTION] Answer: [ANSWER]"

Figure 5.1: Template used to prompt a generative LLM M for a ranking of the codes for a passage. [AVAILABLE_CODES] is a placeholder for the list of codes from $\mathcal{C}$ to rank for an instance $(q, t)_i$, [QUESTION] is a placeholder for the previous question q and [ANSWER] is a placeholder for the text span t highlighted by one of the coders.

via an exact-match search and the order in which the codes are generated is used as their predicted rank. Furthermore, the generation "predicts" the novel code either if M generates the string "None of the above" first (which is included in the prompt template) or if no exact matches are found in its generation.

6

Experiments and Results

We investigate our approaches to modeling the QCS task in order to determine the ability of current NLP-based methods to allow for a more user-oriented approach to qualitative coding. We discuss the experimental setup to test our modeling paradigms in Section 6.1, the ways in which we evaluate our experiments in Section 6.2 and the results and analysis of these results in Sections 6.3 and 6.4 respectively.

6.1 Experimental Setup

To experiment with each of the paradigms presented in the previous section, we sort `CVDQuoding` by the order in which they were originally annotated which we refer to as annotation-time, group it by coder and consider it at different possible points of data saturation. More specifically, for each coder, we consider training on $\{d_1, \ldots, d_K\}$ and testing on $\{d_{K+1}, \ldots, d_{15}\}$ for $K = 1 \ldots 14$ where d_i is the i^{th} annotated transcript in annotation-time. We reserve 20% of each training set for validation. We summarize our workflow in Figure 6.1.

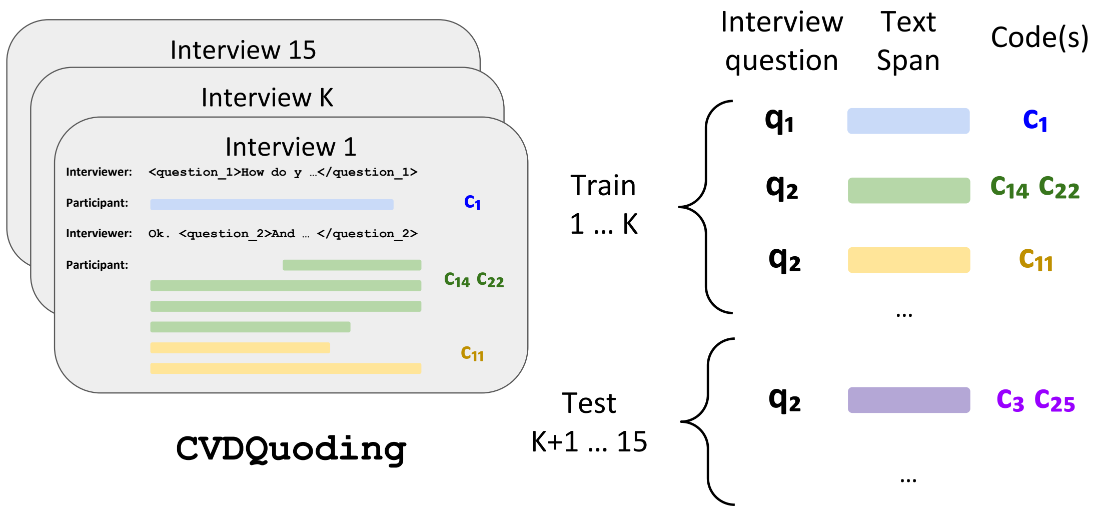


Figure 6.1: Visualization of the data formatting we apply to the raw annotations from `CVDQuoding` to run our experiments with our different modeling paradigms.

We train and test models from each of our three modeling paradigms and for all of the configurations discussed above. For the classification paradigm, we use $|\mathcal{C}|+1$ SVM classifiers as well as a single DistilBERT (Sanh et al., 2020) classifier with a multi-label sequence classification head. We use DistilBERT due to the computational costs of hyperparameter tuning over $2 \times 14 = 28$ individual datasets. For the information-retrieval paradigm, we use DistilBERT as the encoder for the bi-encoder architecture. For the cross-encoder, we experiment with both DistilBert and ConvBERT (Jiang et al., 2021b). We use ConvBERT based on the intuition that text spans often contain phrases which are lexically similar to a code's description (e.g., in-vivo codes). If this is the case, then ConvBERT's span-based kernels may be better suited at soft matching a code's description in a text span than a fully attention-based masked language model like DistilBERT. Moreover, ConvBERT has computational costs (e.g., GPU memory requirements) in the same order of magnitude as DistilBERT. Finally, for our zero-shot prompting paradigm, we use OpenAI's GPT-3.5 Turbo (Brown et al., 2020) accessible through its API[1]. Additional hyperparameter configurations and training details can be found in Appendix A.3.

In addition to the models discussed above, we use an information-retrieval baseline commonly used in neural-retriever papers. We use the Okapi BM25 retriever (Trotman et al., 2014) to compute scores between each code and instance (q, t). In addition, we place a logistic regression model on top of the vector of scores to make the novel class prediction.

6.2 Evaluation Methodology

We assess how well the models from each of our modeling paradigms do on the QCS task by subjecting them to two distinct rounds of evaluations. In the first round of evaluations discussed in Section 6.2.1, we evaluate how well our systems reproduce the coding patterns of the qualitative coders whose annotations were used to create `CVDQuoding`. For instance, to evaluate performance on the main code ranking task, we measure how high our systems rank the codes originally assigned to the text spans in the test set. We refer to this first round of evaluations as the "original annotations" evaluation as the outputs of our systems are exclusively compared to the original annotations of the qualitative coders. In the second set of evaluations discussed in Section 6.2.2, we evaluate how well our systems provide *relevant* code suggestions. To do so, we conduct a human evaluation in which a human evaluator is presented with the rankings of our systems and is asked to determine which of the most highly ranked codes are in fact relevant with respect to the text span. This second round of evaluations is necessary because it is common for text spans to have several alternative and relevant code assignments which the original coders may not have considered. We refer to this round of evaluation as the "human" evaluation as additional human annotations are sought in order to quantify the performance

[1] https://platform.openai.com/docs/api-reference

of systems. We present a visualization of our two-phase evaluation methodology in Figure 6.2.

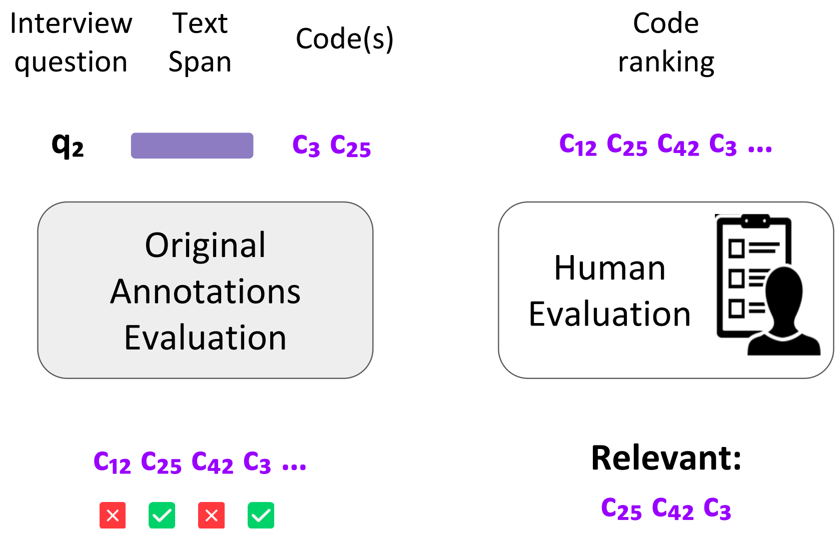


Figure 6.2: The original annotations evaluation relies on the codes assigned to the test text spans in `CVDQuoding` while the human evaluation asks judges to determine which of the top-4 suggested codes are relevant (for a sample of test instances).

6.2.1 Original Annotations Evaluation

We evaluate the performance of our modeling paradigms in solving QCS's main ranking and novel code detection subtask with respect to the original annotations using several automatic metrics. For the main ranking task, we compute the mean reciprocal rank (MRR) score and a soft normalized discounted cumulative gain at k (sNDCG@k) score. For the novel code detection subtask, we compute macro and micro F_1 scores.

To compute the MRR, we consider the set of true codes, C_i^{true}, assigned to a text span i as well as the predicted rank, $\texttt{predicted\_rank}_i : \mathcal{C} \rightarrow \{1, \ldots, |C|\}$, assigned to each code $c \in \mathcal{C}$ for text span i by one of our systems. In this computation, we exclude the novel code as we are only interested in a system's ability to suggest previously assigned codes. The reciprocal rank, RR_i, for a passage i is computed as the maximum predicted reciprocal rank across all true codes (recall that a text span may be assigned multiple codes). That is,

$$\mathrm{RR}_i = \max_{c \in C_i^{\text{true}}} \frac{1}{\texttt{predicted\_rank}_i(c)}$$

The MRR is then computed by averaging across all test instances i with $|C_i^{\text{true}}| \geq 1$.

To compute the sNDCG metric, we approximate a suggested code's c relevance score for a text span i, $rel_i(c)$, in order to carry out the standard NDCG computation. The relevance score, $rel_i(c)$, is computed by using the BERTScore (Zhang et al., 2019) as the approximation for the affinity between suggested and true codes. More specifically, the relevance score $rel_i(c)$, for a code $c \in \mathcal{C}$ and a text span i is

$$rel_i(c) = \max_{c' \in C_i^{\text{true}}} \mathrm{BERTScore}(c, c')$$

Once all the relevance scores are computed, we use them in the standard NDCG computation. To do so, we sort the list of relevance factors $(rel_i(c) : c \in \mathcal{C})$ to compute the `true_rank_scores` list for a text span i. In addition, the `predicted_rank_scores` for a text span i is computed as the list of relevance factors sorted by the model's scores for each code. We can then compute the sNDCG score for the rankings made by our system for text span i as

$$\begin{aligned}
\mathrm{sNDCG}_i &= \frac{\mathrm{sDCG}_i}{\mathrm{sIDCG}_i} \\
\mathrm{sIDCG}_i &= \sum_{i \in 2 \ldots |\mathcal{C}|+1} \frac{\texttt{true\_rank\_scores}[i]}{\log(i)} \\
\mathrm{sDCG}_i &= \sum_{i \in 2 \ldots |\mathcal{C}|+1} \frac{\texttt{predicted\_rank\_scores}[i]}{\log(i)}
\end{aligned}$$

Finally, we add a cutoff k to sNDCG as is usually done in standard NDCG to account only for the top k results creating sNDCG@k. For our purposes, we use $k = 4$ as the `CVDQuoding` dataset has at most 4 code assignments per text span.

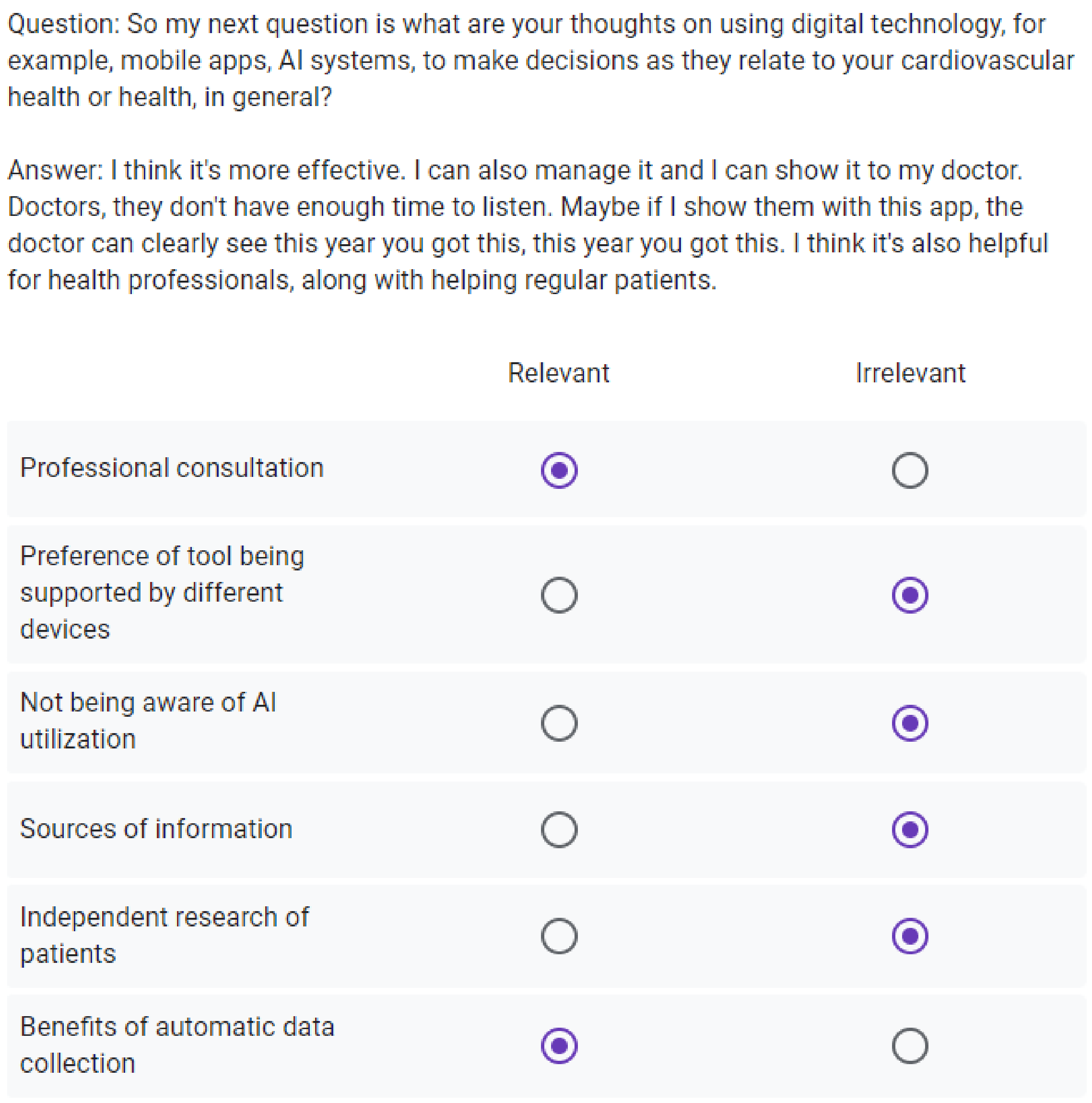
6.2.2 Human Evaluation

In addition to using `CVDQuoding`'s original annotations to compute automatic metrics, we conduct a human evaluation to gather additional annotations to directly ascertain the relevance of our systems' code suggestions. As discussed previously, this additional evaluation step is necessary as automatic metrics computed using the original annotations may not handle well the possibility that multiple codes could fit a text span. As a result, in this section, we first discuss the setup of this human evaluation and then detail the automatic metrics used to measure our systems' performance relative to the newly gathered annotations.

6.2.2.1 Evaluation Setup

To conduct this additional human evaluation, we hire two human evaluators, both with experience in qualitative research, and ask them to judge the relevance of our systems' code suggestions. In particular, we randomly sample 32 instances from the annotations of coders 1 and 2 at the same point of data saturation of $K = 10$ and assign a human evaluator to each sample. We extract the top-4 suggestions of each model described in Section 6.1[2] and ask the evaluator to judge whether each suggestion is "Relevant" or "Irrelevant" based on the question and the text span. Again, we use the top-4 suggestions because, in the `CVDQuoding` dataset, there are at most 4 codes which are assigned to the same text span. We randomly shuffle the selected codes and remove any duplicates. In addition, we always included the codes from the original coding annotations to avoid having the evaluator mark all codes as irrelevant. An example of a question and its corresponding highlighted text span from coder 1's annotations as well as a list of codes (truncated due to space constraints) used to gather human relevance judgments is shown in Figure 6.3.

[2]We do not consider the Okapi BM25 baseline suggestions due to their poor quality.


Question: So my next question is what are your thoughts on using digital technology, for example, mobile apps, AI systems, to make decisions as they relate to your cardiovascular health or health, in general?

Answer: I think it's more effective. I can also manage it and I can show it to my doctor. Doctors, they don't have enough time to listen. Maybe if I show them with this app, the doctor can clearly see this year you got this, this year you got this. I think it's also helpful for health professionals, along with helping regular patients.

	Relevant	Irrelevant
Professional consultation	●	○
Preference of tool being supported by different devices	○	●
Not being aware of AI utilization	○	●
Sources of information	○	●
Independent research of patients	○	●
Benefits of automatic data collection	●	○

Figure 6.3: An example of a question and a text span from coder 1's annotations shown to the human evaluator. We truncate the list of codes shown due to space constraints.

6.2.2.2 Automatic Metrics

We use the annotations from the human evaluation to recompute the rank-based automatic metrics presented in Section 6.2.1. That is, for a sampled instance i, C^i_{true} becomes the set of codes selected as “Relevant” by the evaluator. In addition, we also compute the precision at 4 (P@4) of each system using the human evaluator’s annotations. That is, let $N_{@4}$ be the number of relevant codes, as indicated by the human evaluators’ annotations, in the first 4 suggested codes. Then the $P@4_i$ for a text span i is computed as

$$\text{P@4}_i = \frac{N_{@4}}{4}$$

We then average the instance-level precision scores over all 32 sampled test instances to get $P@4$.

6.3 Results

We present the results of our experiments for both rounds of evaluations.

6.3.1 Original Annotations

Using the original annotations as the gold standard labels, we present the MRR and sNDCG@4 scores for the main ranking task of QCS as well as the macro and micro F_1 for the novel code detection subtask. In particular, we show the MRR and sNDCG@4 scores for the main code ranking task at $K = 10$ for coders 1 and 2 in Table 6.1 and the F_1 scores for the novel code prediction task at $K = 5$ in Table 6.2. Additional results for all points of data saturation, presented using scatter plots, can be found in Appendix A.4.

For the main ranking task, we observe that the information-retrieval paradigm, modeled through the ConvBERT cross-encoder, and the zero-shot prompting paradigm, modeled through GPT-3.5, achieve the highest MRR and sNDCG@4 scores across

	MRR		sNDCG@4	
	Coder 1	Coder 2	Coder 1	Coder 2
Okapi BM25	0.15	0.53	0.70	0.79
SVM	0.51	0.75	0.80	0.85
DistilBERT	0.55	0.75	0.70	0.79
Bi-Encoder	0.48	0.74	0.80	0.86
Cross-Encoder (DistilBERT)	0.55	0.64	0.81	0.82
Cross-Encoder (ConvBERT)	**0.59**	**0.77**	**0.83**	**0.86**
GPT-3.5	0.57	0.73	0.74	0.83

Table 6.1: Results of the models' performance on QCS' main ranking task. The MRR and sNDCG@4 scores are computed based on the original annotations from the `CVDQuoding` dataset and using $K = 10$ as the point of data saturation.

both coders for most data saturation points. While GPT-3.5 achieves the highest MRR in most cases for coder 1 (11 out of 14), ConvBERT obtains the largest number of maximal MRR scores for coder 2 (7 out of 14). In addition, the sNDCG@4 scores are dominated by the ConvBERT model. This latter result may be a side-effect of using a BERT-derived metric to evaluate BERT-based models. For the novel code detection subtask, performances are relatively poor with the Okapi BM25 baseline model achieving the highest macro and micro F_1. These poor novel classification scores are reflective of the inherent difficulty of detecting *rare* codes and consistent with previous work (Crowston et al., 2012; Grandeit et al., 2020).

6.3.2 Human Evaluation

Using the human evaluation annotations as the gold standard labels, we present the results for the same rank-based metrics and contrast them with the results in the previous section. Consistent with the automatic metrics computed using the original annotations, we observe that the information-retrieval and zero-shot prompting paradigm achieve the highest MRR and sNDCG@4 (Table 6.3). In addition, we observe relatively saturated sNDCG@4 scores. This saturation is a result of the cu-

	Macro F1		Micro F1	
	Coder 1	Coder 2	Coder 1	Coder 2
Okapi BM25	**0.56**	**0.57**	**0.59**	**0.80**
SVM	0.32	0.30	0.44	0.30
DistilBERT	0.55	0.45	**0.59**	**0.80**
Bi-Encoder	0.36	0.45	0.57	**0.80**
Cross-Encoder (DistilBERT)	0.51	0.46	**0.59**	0.78
Cross-Encoder (ConvBERT)	0.36	0.57	0.57	0.79
GPT-3.5	0.41	0.44	0.53	**0.80**

Table 6.2: Novel class detection subtask results. Macro and micro F_1 scores are computed using $K = 5$.

mulative nature of the NDCG computation combined with the increase in the number of relevant codes identified by the human evaluators. Indeed, while the human evaluators have a recall with respect to the original annotations of 0.94 and 0.95 for coders 1 and 2 respectively, they identify, on average, 3 times more relevant codes than in the original annotations.

In addition to the saturation of the sNDCG@4 metric and the increase in number of relevant codes, we observe a statistically significant jump between the MRR and sNDCG@4 scores computed on the sample of 32 text spans using the original annotations and the human evaluation annotations. On average, the human-evaluation-derived MRR is 66% and 34% higher than the MRR computed using the original annotations for coders 1 and 2, respectively. In fact, in the case of coder 2, the recomputed MRR for GPT-3.5 is 1. This jump is less drastic for the sNDCG@4 metric with relative increases of 40% and 19% for coders 1 and 2 respectively. Thus, these results suggest that additional human evaluations are necessary as they provide interpretations for codes which were highly ranked by our systems but which the qualitative coders may not have originally considered in their code assignments. This latter claim is also supported by the relatively high P@4 scores across all models - above 0.5 in all cases - indicating that, on average, several previously assigned codes may be relevant

	MRR		sNDCG@4		P@4	
	Coder 1	Coder 2	Coder 1	Coder 2	Coder 1	Coder 2
SVM	0.80	0.93	0.970	0.978	0.40	0.55
DistilBERT	0.72	0.91	0.969	0.975	0.41	0.65
Bi-Encoder	0.86	0.90	0.972	0.979	0.50	0.66
Cross-Encoder (DistilBERT)	0.78	0.93	0.965	0.982	0.46	**0.68**
Cross-Encoder (ConvBERT)	**0.87**	0.95	**0.973**	0.987	**0.51**	0.56
GPT-3.5	0.74	**1.00**	**0.973**	**0.989**	0.32	0.53

Table 6.3: MRR, sNDCG@4 and P@4 scores computed from the annotations collected during the human evaluation.

for a newly highlighted passage.

6.4 Analysis

6.4.1 Model Comparison

We further discuss the differences in model performance by comparing the predictions made by the models from each of the three modeling paradigms. In particular, we show sample model outputs for the SVM, the Cross-Encoder (implemented using ConvBERT) and GPT-3.5 along with the original code annotations and the annotations made during the additional human evaluation for both coders in Tables 6.4 and 6.5.

We characterize the differences in model performance with respect to the original coder annotations as well as the additional human evaluation annotations. On one hand, when considering the original coder annotations, we observe that the SVM and the CrossEncoder implemented with the ConvBERT consistently rank the original codes higher than GPT-3.5. This discrepancy in performance, which at first could seem surprising given GPT-3.5's superior capacity to generalize, is observed for both coders and is explained by the fact that both the SVM and the CrossEncoder were fine-tuned and, thus, have already been exposed to the coding style of both coders. As a result, while GPT-3.5 is able to make relevant code suggestions, the fine-tuned models are better at retrieving the coder's preferred codes which in principle should already be relevant. For instance, because the fine-tuned models have been exposed to terms such as "difficulty" and "Xi-Care" co-occuring with the code "Level of difficulty of integrating Xi care tool", they perform better when similar examples occur at test time (Table 6.4). In contrast, GPT-3.5 has not been exposed to these training examples and therefore proposes more generic codes such as "Additional comments about app related support"[3]. On the other hand, when considering the additional annotations made by human evaluators, this discrepancy in performance is more nuanced. The SVM and CrossEncoder models still perform better at ranking relevant codes for coder 1, however, for coder 2, this trend is reversed and GPT-3.5 *always* ranks relevant codes first. We believe that GPT-3.5's improved performance under this latter evaluation setting is due to the interaction between its generalizability capacities and the fact that many alternative relevant codes may exist for a given passage. In this setting, when a test time example does not resemble any training examples, GPT-3.5 is better suited at suggesting relevant codes than the fine-tuned models which, by replicating the coder's coding style, might provide unrelated code suggestions.

Effectively, the evaluation-setting-based differences in performance of the models suggest that the modeling paradigms, fine-tuning, via classification or information-retrieval, and zero-shotting, via prompting, may serve different and complementary purposes under QCS. Using a classification or information-retrieval approach to fine-tune models for QCS is necessary to create a system which captures the coding style of a coder and therefore provides personalized code suggestions. In contrast, using prompting to zero-shot QCS's main ranking task serves as a way to provide the coder with alternative points of view which they may not have otherwise considered. Together, the code suggestions of these two systems could provide a coder with a way to efficiently sift through possible code assignments while considering "unconventional"

[3]"app" is short for application

code alternatives.

6.4.2 Connection with the Research Questions

We discuss the implications of our results in relation to our two initial research questions. To do so, we first discuss the connections we observe between our empirical results and the three properties of qualitative coding which we identified at the beginning of this thesis and integrated into our task definition. In addition, we discuss the implications of our results on the effectiveness of current NLP techniques at handling these properties and, by extension, this human-centric approach to qualitative coding.

Firstly, across both coders and all models, we observe a steady rise in our systems' ability to make code suggestions consistent with the original annotations of `CVDQuoding` (See MRR and sNDCG@4 scatter plots through annotation-time in Appendix A.4). This poor performance for small values of K is consistent with inductive coding which, for the first documents, is exploratory and unpredictable by nature. In fact, the moment at which the rise in automatic metrics (MRR, sNDCG) plateaus may be a helpful signal to identify when a coder has reached their personal point of *data saturation* and, thus, when code suggestions could start to be beneficial. Secondly, across all saturation points and for both coders, we observe relatively poor performance in detecting novel codes. This difficulty suggests that the "catch-all" bucket method we used in this thesis is inappropriate and that more sophisticated representations of novelty need to be learned to properly detect *rare codes.* Finally, we notice that, on average, our systems' MRR and the sNDCG@4 calculated using coder 2's annotations are 58% and 7% higher respectively than when calculated using coder 1's annotations. This result is natural given coder 2's *annotation style* of assigning high-level codes to larger text spans (i.e., lumping) is much more amenable to machine learning techniques. Thus, in the future, methods that are able to identify annotation styles and incorporate them in their modeling, for example through some

Question/Text Span	Top-4 Code Suggestions
QUESTION: "And what do you think are some of the challenges or needs in terms of managing cardiovascular diseases from your perspective? Do you find there's a lack of resources or guidance on how to maintain cardiovascular health?" ANSWER: "I mean, I think the doctor said losing weight is a good idea. Stuff like that. I do exercise a lot already. But no, other than that, I'm not sure what what to do in terms of managing it."	*SVM:* Lack of knowledge, CVD misconceptions, **Life-style modification**, Importance of preventive measures *CrossEncoder:* Importance of preventive measures, **Life-style modification**, Tendency for not seeking medical attention from family doctor, Workout routine and sources *GPT-3.5:* Lack of knowledge, CVD misconceptions, Interest in the use of the app, Level of interest in weight tracking feature
QUESTION: "From one to five, with one being very difficult to five being very easy, how easy or difficult do you think it will be for you to use the Xi-Care tool in your daily life?" ANSWER: "The question is whether I will be consistent with it. It's not whether the app will be good. The the app will be good. It's whether I will stay engaged with the app that might be a problem. That's a personal thing. If it's engaging and I feel it's making a difference, or I am engaged in it, then I will probably use it. I do for other apps. Can I integrate it in? Absolutely. Will I be consistent with it? That's a personal thing."	*SVM:* **Level of difficulty of integrating Xi care tool**, Increasing trust in the tool, Additional comments about explainability, Needs to assure data confidentiality and security *CrossEncoder:* Additional comments about facilitating integration of Xi-Care tool, **Level of difficulty of integrating Xi care tool**, Visualization of progress, Level of trust in recommendations *GPT-3.5:* Additional comments about app related support, Need to know where data is stored, Automatic capturing apps and devices, Favouring step-count features over workouts
QUESTION: "What would facilitate its integration into your daily life activities? So how could we make it so you're able to use them more often?" ANSWER: "I think it has to be a quick tool. In other words, it has to be seamless, it's hard, that's why that's why I would say forget the computer."	*SVM:* **Additional comments about facilitating integration of Xi-Care tool**, App prompts, Increasing trust in the tool, Level of difficulty of integrating Xi care tool *CrossEncoder:* **Additional comments about facilitating integration of Xi-Care tool**, Enthusiastic about technology, Integration of inter-related tools, Level of difficulty of integrating Xi care tool *GPT-3.5:* Preference of tool being supported by different devices, Hectic daily life and responsibilities, Integration of inter-related tools, Accessibility of the tool to target populations

Table 6.4: Top-4 code suggestions for the SVM, CrossEncoder and GPT-3.5 model (as ordered by the model) for 3 question-text span pairs sampled from the $K = 10$ split for Coder 1. The codes **in bold** are the ones the coder originally assigned to the passages while the underlined codes are the ones that were determined to be relevant by the human evaluators.

Question/Text Span	Top-4 Code Suggestions
QUESTION: "Do you see any challenges with the integration of the tool in your daily life in terms of ethics? Keep in mind the table that we presented earlier. Do you see any type of challenges when you use or you integrate this tool in your life in terms of ethics?" ANSWER: "I am not sure. You mean like my data being shared, in that sense like? I'm not that concerned about that no."	*SVM:* Ethical Concerns, Transparency, **Data Confidentiality**, Technical Support *CrossEncoder:* Ethical Concerns, **Data Confidentiality**, Challenges, Trust and Reliability *GPT-3.5:* **Data Confidentiality**, Technical Support, Diet Recommendations, CVD Presentation feature
QUESTION: "Would you like to be able to follow your progress and receive push notification through Xi-Care tool?" ANSWER: "Definitely. Yes."	*SVM:* Data Monitoring, **Notification Preferences**, Transparency, Trust and Reliability *CrossEncoder:* **Notification Preferences**, Data Monitoring, Cardiovascular Health Information, Health Measures Taken *GPT-3.5:* **Notification Preferences**, Health Measures Taken, Tool Automaticity and Data Collection, Diet Recommendations
QUESTION: "And in terms of educational modules on cardiovascular health, how helpful would that be in your opinion?" ANSWER: "I was just basing that on some websites like Mayo clinic where you can go in and say what your symptoms are, click on parts of the body, and it narrows it down to a few different options and then it gives you all the information about each option."	*SVM:* Educational Models, Cardiovascular Health Information, Decisions Made, Trust and Reliability *CrossEncoder:* Educational Models, Cardiovascular Health Information, Technology for CVD care, Accessibility *GPT-3.5:* Educational Models, Technical Support, Diet Recommendations, CVD Presentation

Table 6.5: Top-4 code suggestions for the SVM, CrossEncoder and GPT-3.5 model (as ordered by the model) for 3 question-text span pairs sampled from the $K = 10$ split for Coder 2. The codes **in bold** are the ones the coder originally assigned to the passages while the underlined codes are the ones that were determined to be relevant by the human evaluators.

prior, may be more suitable at handling drastic differences in annotations styles.

In addition, the ranked-based metrics computed using human judgments point to a promising future for the downstream viability of QCS systems. The recomputed MRR and P@4 metrics show that not only are our systems able to correctly rank relevant codes, but they are also likely to highly rank more than one relevant code, potentially forcing the coder to further reflect on their coding decisions without needing to consider irrelevant codes. This suggests that this approach to qualitative coding is quite promising and, given that it is supported by users' desires, may be able to assist researchers in their qualitative coding.

7 Conclusion

In this thesis, we approached qualitative coding in a novel user-oriented manner and showed that this approach gives rise to technical subtleties related to automatic coding which have not been previously investigated. These subtleties, which were directly integrated into the QCS task we studied, included the importance of considering a coder's personal point of data saturation, the difficulty of detecting rare codes and the necessity to avoid homogenizing different annotation styles. We showed that each of these properties is reflected in both the `CVDQuoding` dataset we introduced and the experimental results of our modeling of QCS. Lastly, our human evaluation showed that solely relying on automatic metrics computed with respect to the original annotations made by the qualitative coders is insufficient and may underestimate systems' performance. The results derived from our human evaluation showed that QCS systems can consistently provide several relevant code suggestions and, thus, that this human-centric approach to qualitative coding may be able to truly *assist* researchers in their study of textual corpora.

7.1 Limitations and Future Work

This work includes a few limitations which we leave as future work. Firstly, this work does not investigate the use of our QCS systems in an applied setting. This lack of downstream experiments prevents us from determining the full impact of automatically providing code suggestions to qualitative coders. For instance, an automatic QCS system may remove more agency from the coder than anticipated if they begin to blindly trust suggestions. In addition, our human evaluation, which was meant to act as a proxy for downstream performance, makes assumptions which do not directly align with how qualitative coding is conducted. This misalignment stems from the limited context given to the evaluators to judge the relevance of suggestions. In particular, since the evaluators did not code any of the transcripts themselves, they may have lacked some of the intuition needed to make code relevance judgments. As a result, future work should consider exploring more holistic methods of evaluating qualitative code suggestions. Secondly, despite our best efforts, our work falls short of investigating all possible technical avenues to solving QCS. For instance, we do not investigate fine-tuning sequence-to-sequence models as the preliminary experiments we ran showed poor performance. In addition, we do not explore the full range of abilities offered to us by LLMs such as GPT-3.5. For instance, using GPT-3.5 to generate synthetic annotations may have helped the performance of our larger neural-based models which are known to perform poorly in data scarce settings. Finally, the scope of `CVDQuoding` dataset prevented us from exploring the QCS task on other types of textual corpora (e.g., questionnaires, focus groups) and on other language domain (i.e., beyond the CVD-related vocabulary). We believe that the creation of more datasets is needed to truly understand the applicability of the QCS task to the qualitative coding endeavour.

Bibliography

Anderson, C. (2010). Presenting and evaluating qualitative research. *American journal of pharmaceutical education*, 74(8).

Bakharia, A. (2014). *Interactive content analysis: evaluating interactive variants of non-negative matrix factorisation and latent Dirichlet allocation as qualitative content analysis aids*. PhD thesis, Queensland University of Technology.

Bakharia, A., Bruza, P., Watters, J., Narayan, B., and Sitbon, L. (2016). Interactive topic modeling for aiding qualitative content analysis. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 213–222.

Bengtsson, M. (2016). How to plan and perform a qualitative study using content analysis. *NursingPlus open*, 2:8–14.

Bousbiat, I. C., Roland, G., Shahram, Y., Rodriguez, C., Pluye, P., Légaré, J., Bergman, H., Vedel, I., Bisleri, G., Gagnon, M., and Abbasgholizadeh-Rahimi, S. (2022). *"CVD Prevention among Women in Primary Care Using AI: Work-in-Progress"*. Family Medicine Forum/Forum en médecine familiale, Toronto, Ontario.

Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Chapman, A., Hadfield, M., and Chapman, C. (2015). Qualitative research in healthcare: an introduction to grounded theory using thematic analysis. *Journal of the Royal College of Physicians of Edinburgh*, 45(3):201–205.

Chen, N.-C., Drouhard, M., Kocielnik, R., Suh, J., and Aragon, C. R. (2018). Using machine learning to support qualitative coding in social science: Shifting the focus to ambiguity. *ACM Trans. Interact. Intell. Syst.*, 8(2).

Chen, N.-c., Kocielnik, R., Drouhard, M., Peña-Araya, V., Suh, J., Cen, K., Zheng, X., Aragon, C. R., and Peña-Araya, V. (2016). Challenges of applying machine learning to qualitative coding. In *ACM SIGCHI Workshop on Human-Centered Machine Learning*.

Crowston, K., Allen, E., and Heckman, R. (2012). Using natural language processing technology for qualitative data analysis. *International Journal of Social Research Methodology*, 15(6):523–543.

Crowston, K., Liu, X., and Allen, E. E. (2010). Machine learning and rule-based automated coding of qualitative data. *proceedings of the American Society for Information Science and Technology*, 47(1):1–2.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Drisko, J. W. and Maschi, T. (2016). *Content analysis*. Pocket Guide to Social Work Re.

Drouhard, M., Chen, N.-C., Suh, J., Kocielnik, R., Pena-Araya, V., Cen, K., Zheng, X., and Aragon, C. R. (2017). Aeonium: Visual analytics to support collaborative qualitative coding. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 220–229. IEEE.

Elliott, V. (2018). Thinking about the coding process in qualitative data analysis. *The Qualitative Report*, 23(11):2850–2861.

Feuston, J. L. and Brubaker, J. R. (2021). Putting tools in their place: The role of time and perspective in human-ai collaboration for qualitative analysis. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–25.

Fielding, N. G. and Lee, R. M. (2002). New patterns in the adoption and use of qualitative software. *Field Methods*, 14(2):197–216.

Gibbs, G. R. (2007). Thematic coding and categorizing. *Analyzing qualitative data*, 703:38–56.

Górska, S., Forsyth, K., and Maciver, D. (2018). Living with dementia: a meta-synthesis of qualitative research on the lived experience. *The Gerontologist*, 58(3):e180–e196.

Gough, S. and Scott, W. (2000). Exploring the purposes of qualitative data coding in educational enquiry: Insights from recent research. *Educational Studies*, 26(3):339–354.

Grandeit, P., Haberkern, C., Lang, M., Albrecht, J., and Lehmann, R. (2020). Using BERT for Qualitative Content Analysis in Psychosocial Online Counseling. In *Proceedings of the Fourth Workshop on Natural Language Processing and Compu-*

tational Social Science, pages 11–23, Online. Association for Computational Linguistics.

Grantham, A., Robinson, E. E., and Chapman, D. (2015). “that truly meant a lot to me”: A qualitative examination of meaningful faculty-student interactions. *College Teaching*, 63(3):125–132.

Grimmer, J. and Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297.

Hsieh, H.-F. and Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative health research*, 15(9):1277–1288.

Jiang, J. A., Wade, K., Fiesler, C., and Brubaker, J. R. (2021a). Supporting serendipity: Opportunities and challenges for human-ai collaboration in qualitative analysis. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–23.

Jiang, Z., Yu, W., Zhou, D., Chen, Y., Feng, J., and Yan, S. (2021b). Convbert: Improving bert with span-based dynamic convolution.

Kaufmann, A., Barcomb, A., and Riehle, D. (2020). Supporting interview analysis with autocoding. In *HICSS*, pages 1–10.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4).

Liew, J. S. Y., McCracken, N., Zhou, S., and Crowston, K. (2014). Optimizing Features in Active Machine Learning for Complex Qualitative Content Analysis. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 44–48, Baltimore, MD, USA. Association for Computational Linguistics.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Loslever, P., Guidini Gonçalves, T., de Oliveira, K. M., and Kolski, C. (2019). Using fuzzy coding with qualitative data: example with subjective data in human-computer interaction. *TheoreTical issues in ergonomics science*, 20(4):459–488.

Macnamara, J. (2006). The fork in the road of media and communication theory and practice. *4th Annual Summit on Measurement*, pages 1–12.

Marathe, M. and Toyama, K. (2018). Semi-automated coding for qualitative research: A user-centered inquiry and initial prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA. Association for Computing Machinery.

McCracken, N., Suet Yan Yan, J., and Crowston, K. (2014). Design of an active learning system with human correction for content analysis. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 59–62.

Miles, M., Huberman, A., and Saldana, J. (2019). *Qualitative Data Analysis: A Methods Sourcebook*. SAGE Publications.

Muller, M., Guha, S., Baumer, E. P., Mimno, D., and Shami, N. S. (2016). Machine learning and grounded theory method: convergence, divergence, and combination. In *Proceedings of the 2016 ACM International Conference on Supporting Group Work*, pages 3–8.

Patton, M. Q. (2014). *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Rietz, T. and Maedche, A. (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, New York, NY, USA. Association for Computing Machinery.

Roberts, K., Dowell, A., and Nie, J.-B. (2019). Attempting rigour and replicability in thematic analysis of qualitative research data; a case study of codebook development. *BMC medical research methodology*, 19:1–8.

Saldaña, J. (2014). Coding and analysis strategies. In *The Oxford Handbook of Qualitative Research*, pages 581–598.

Saldaña, J. (2021). The coding manual for qualitative researchers. *The coding manual for qualitative researchers*, pages 1–440.

Sandhu, A., Vamvakas, K., Gagnon, M.-P., Yousefi, S., Bergman, H., Grad, R., Pluye, P., Vedel, I., Rodriguez, C., and Abbasgolizadeh-Rahimi, S. (2023). *AI for CVD Management: Primary care providers' perspectives.* Family Medicine Forum/Forum en médecine familiale, Montreal, Quebec.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Santiago-Rivera, A. L., Altarriba, J., Poll, N., Gonzalez-Miller, N., and Cragun, C. (2009). Therapists' views on working with bilingual spanish–english speaking clients: A qualitative investigation. *Professional Psychology: Research and Practice*, 40(5):436.

Saunders, B., Sim, J., Kingstone, T., Baker, S., Waterfield, J., Bartlam, B., Burroughs, H., and Jinks, C. (2018). Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & quantity*, 52:1893–1907.

Scharkow, M. (2013). Thematic content analysis using supervised machine learning: An empirical evaluation using German online news. *Quality & Quantity*, 47(2):761–773.

Smith, B. and McGannon, K. R. (2018). Developing rigor in qualitative research: Problems and opportunities within sport and exercise psychology. *International review of sport and exercise psychology*, 11(1):101–121.

Terry, G., Hayfield, N., Clarke, V., and Braun, V. (2017). Thematic analysis. *The SAGE handbook of qualitative research in psychology*, 2:17–37.

Trotman, A., Puurula, A., and Burgess, B. (2014). Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, page 58–65, New York, NY, USA. Association for Computing Machinery.

van Rijnsoever, F. J. (2017). (i can't get no) saturation: a simulation and guidelines for sample sizes in qualitative research. *PloS one*, 12(7):e0181689.

Wiedemann, G. (2013). Opening up to big data: Computer-assisted analysis of textual data in social sciences. *Historical Social Research/Historische Sozialforschung*, pages 332–357.

Xiao, Z., Yuan, X., Liao, Q. V., Abdelghani, R., and Oudeyer, P.-Y. (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*, pages 75–78.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

A

Appendix

A.1 CVDQuoding Questions

Question 1	How do you think cardiovascular diseases are generally described and understood by the public?
Question 2	Have you ever been informed about your cardiovascular health? How?
Question 3	What are the most frequent and important decisions you face related to your cardiovascular health?
Question 4	What is your usual role in making decisions about your cardiovascular health and preventing CVD?
Question 5	What do you think are some challenges and needs in preventing and managing cardiovascular diseases (from your perspective as a woman at risk of CVD)?
Question 6	Have you ever considered a decision support system to help you in any decisions related to your cardiovascular health?
Question 7	What are your thoughts on using digital technology (e.g., mobile apps, AI systems/robots) to make decisions in relation to your cardiovascular health?
Question 8	How would you like us to design and develop this Xi-Care tool that is useful, helpful and effective for women at risk of CVD (e.g. no risks to users)?
Question 9	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in monitoring tools that track health data over time?
Question 10	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in a step-count feature?
Question 11	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in a weight tracking feature?
Question 12	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in educational modules on cardiovascular health?
Question 13	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in guided exercise activities?
Question 14	On a scale of 1 to 5 with 1 being not at all interested and 5 being very interested, what is your level of interest in diet recommendations?
Question 15	Would you like to be able to follow your progress and receive push-notifications through the Xi-Care tool?
Question 16	How difficult/easy do you think it will be for you to integrate the Xi-Care tool into your daily life?
Question 17	To what extent would you trust or rely on the Xi-Care tool to make assessments about your cardiovascular health and prevent and manage CVD?
Question 18	Do you foresee any challenges with integrating the Xi-Care tool in your daily life in terms of ethics? Could you please describe these challenges?
Question 19	In terms of transparency, how important is it for you to be able to understand how the Xi-Care tool works?
Question 20	Is there something else you'd like to add about ethical aspects in regards to the Xi-Care tool that will be empowered by AI (Justice; Non-maleficence; Autonomy; Beneficence; Explicability/Transparency)?

Table A.1: List of questions asked during the interviews of the `CVDQuoding` dataset. *Xi-Care* is the name of the app being proposed to participants of the study to help them control their risks of cardiovascular diseases (CVDs).

A.2 Additional Dataset Details

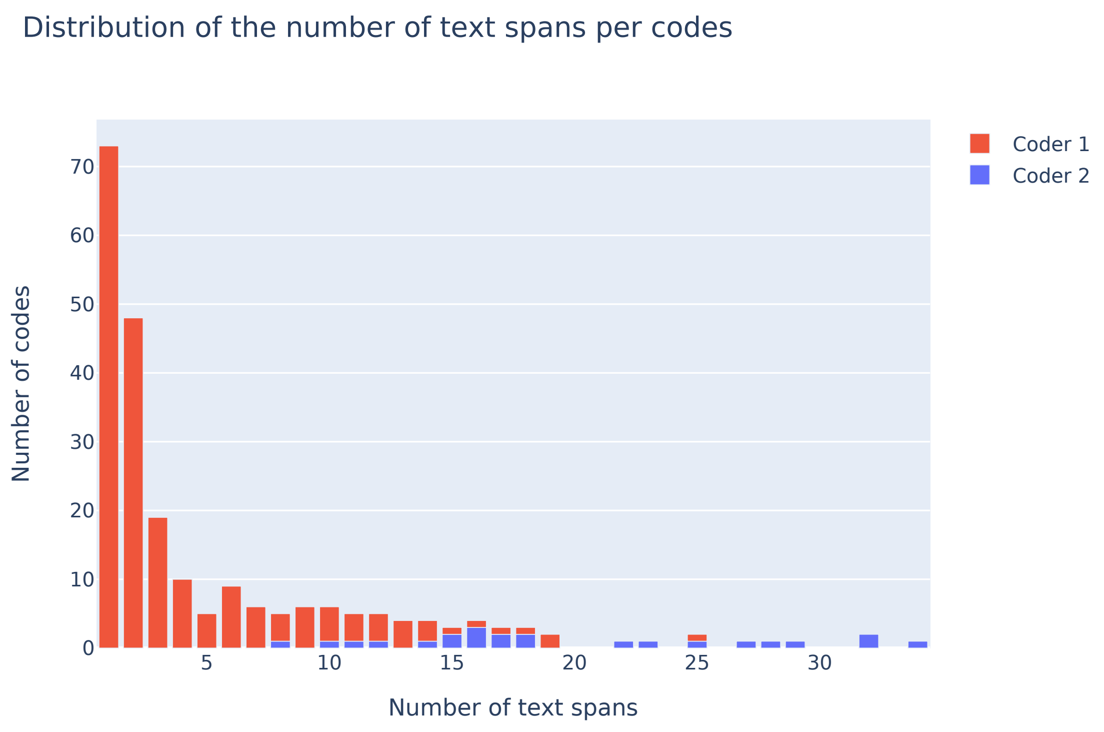


Figure A.1: Frequency of code assignment counts.

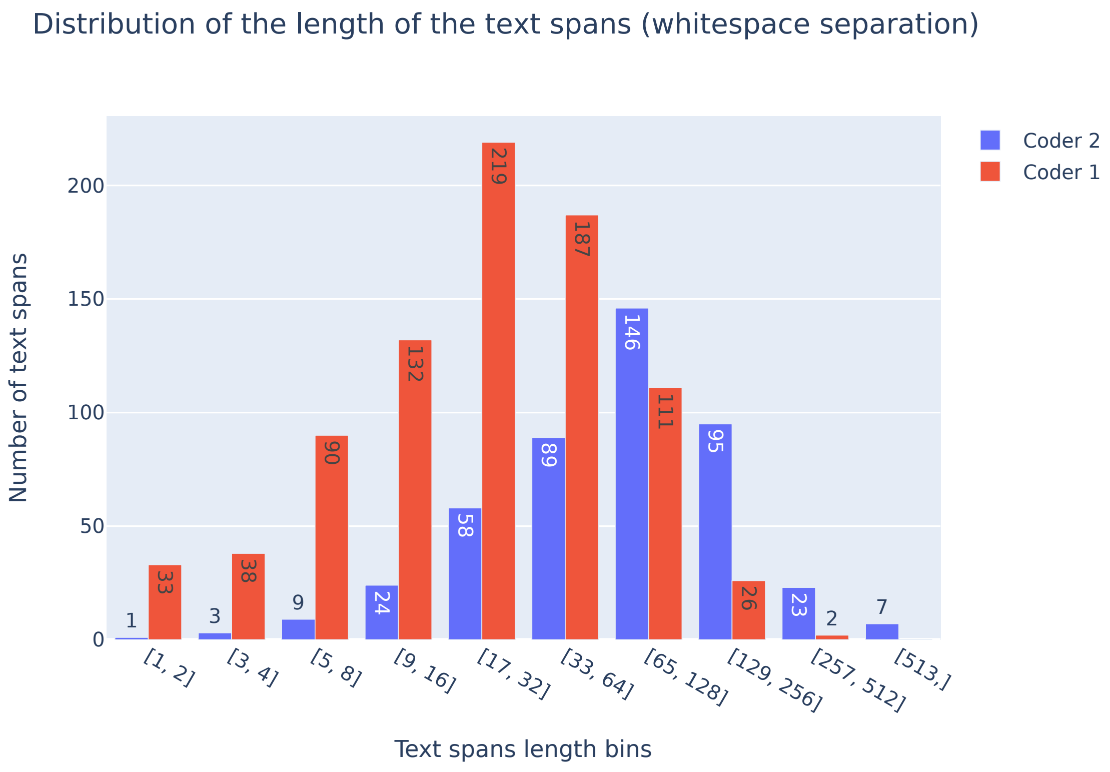


Figure A.2: Distribution of the length of text spans.

Coder	List of codes
Coder 1	Life-style modification, Needs to assure data confidentiality and security, Increasing trust in the tool, Lack of knowledge, Additional comments about diet recommendations, CVD misconceptions, Level of interest in tracking information and prompts, Level of interest in educational modules, Additional comments about educational modules, Level of interest in transparency and explainability, Enthusiastic about technology, App prompts, Frequency of push-notifications, Level of interest in personalized workouts, Additional comments about explainability
Coder 2	Technology for CVD care, Cardiovascular Health Information, Accessibility, Decisions Made, Additional Features Suggested, Understanding of CVD, Challenges, Trust and Reliability, Transparency, Decision Involvement, Weight Tracking, Educational Models, Technical Support, Data Monitoring, Pedometer

Table A.2: List of 15 codes created by coders 1 and 2 sorted by frequency over the entire `CVDQuoding` dataset.

A.3 Hyperparameter Configurations

We provide additional hyperparameter and architecture details for all our models.

A.3.1 Baseline

Okapi-BM25. We use an open-source implementation of the BM25 algorithm[1]. We use the Okapi implementation with its default parameters $k_1 = 1.5, b = 0.75, \varepsilon = 0.25$ and a tokenizer with space separation and lowercasing.

A.3.2 Classification Paradigm

SVM. We use the SVM implementation from sklearn[2]. We transform all inputs to tf-idf features with separate encodings for the question and text span. We use the

[1]https://github.com/dorianbrown/rank_bm25

[2]https://scikit-learn.org/stable/

standard radial basis function kernel and tune the regularization parameter C using the values $\{0.001, 0.01, 0.1, 1, 10\}$. A class weight parameter is computed for each code as well as for the novel code and is inversely proportional to the class frequencies in the training set.

DistilBERT. We use the DistilBERT implementation from Hugging Face[3]. In particular, we use DistilBERT's tokenizer and its DistilBertForSequenceClassification module which accepts a variable number of labels, $|C| + 1$ in this case. We tune the learning rate using values $\{1 \times 10^{-5}, 2 \times 10^{-5}\}$ with a weight decay of 0.01. We use a batch size of 8 and train for 25 epochs with early stopping on the validation loss with a patience of 5 epochs. We also compute a class weight inversely proportional to the class frequencies in the training set and apply it to the cross entropy loss. All our experiments consistently show that using class weights benefits performance.

A.3.3 Information-retrieval Paradigm

Along with hyperparameter details, we provide additional architecture details about both the bi-encoder and cross-encoder.

Bi-Encoder. In the bi-encoder, we create representations for the code c, the text span t and the previous question q with three distinct BERT encoders, E_t, E_q and E_c. In our case, we use DistilBERT as the encoder for all three inputs. We pass each input whose start is truncated to the model's 512 token input limit to its respective encoder and extract its CLS token representation.

$$h_t = E_t(t), h_q = E_q(q), h_c = E_c(c)$$

We use a component-wise max pooling method $\max(\cdot, \cdot)$ to aggregate h_t and h_q together.

$$h_{\text{pool}} = \max(h_t, h_q)$$

[3]https://huggingface.co/

Finally, a cosine similarity is applied between the pooled representation, h_{pool}, and the code representation, h_c, to generate a score. A *class weighted* mean-squared error loss is then computed and backpropagated through the three encoders. For the novel code detection subtask, a linear classifier is placed on top of the vector of encoder scores. For the training of the linear classifier, we use another class weighted cross entropy loss and keep the three encoders frozen.

We tune the learning rate using the values $\{1 \times 10^{-5}, 2 \times 10^{-5}\}$ with a weight decay of 0.01. We use a larger batch size of 32, since training sets now have a size of $|\mathcal{C}| \times N$ where N is the number of coded text spans. We train for 25 epochs with early stopping on the validation loss set with a patience of 5 epochs.

Cross-Encoder. In the cross-encoder, we create representations for the concatenation of codes with the question and the text span. That is, for every code c in the training set, the text span t and the previous question q are each independently concatenated with the code c using the special [SEP] token. Both concatenations are passed to different encoders E_t and E_q to obtain a contextual representation. Truncation is applied to the start of both t and q to comply with the encoder's maximum token input length.

$$h_{t \text{ [SEP] } c} = E_t(t \text{ [SEP] } c)$$
$$h_{q \text{ [SEP] } c} = E_q(q \text{ [SEP] } c)$$

The representations are pooled using a component-wise max pooling method $\max(\cdot, \cdot)$

$$h_{\text{pool}} = \max(h_{t \text{ [SEP] } c}, h_{q \text{ [SEP] } c})$$

We place a classification head on top of h_{pool} identical in architecture to the DistilBERT classification head. We compute a *class weighted* cross entropy loss and backpropagate it through both encoders. For the novel code detection subtask, we a linear classifier is placed on top of the vector of encoder scores, which, in this case, are logits. We use a class weighted cross entropy loss and keep the two encoders frozen.

In this case, we experiment with DistilBERT and ConvBERT. In both cases, we tune the learning rate using the values $\{1 \times 10^{-5}, 2 \times 10^{-5}\}$ with a weight decay of 0.01. We use a batch size of 32 and train for 30 epochs with early stopping on the validation loss set to have a patience of 5 epochs. We train for more epochs because we noticed that the cross-encoders took longer to converge.

A.3.4 Zero-shot Paradigm

We use OpenAI's GPT-3.5 API to generate LLM responses to our zero-shot prompt. In particular, we use the model checkpoint `gpt-3.5-turbo-0301` with a temperature of 1. The question and the text span are truncated to allow for a 64 token generation. At the time of our experiments, this version of GPT-3.5 allowed for 4096 tokens.

A.4 Additional Results

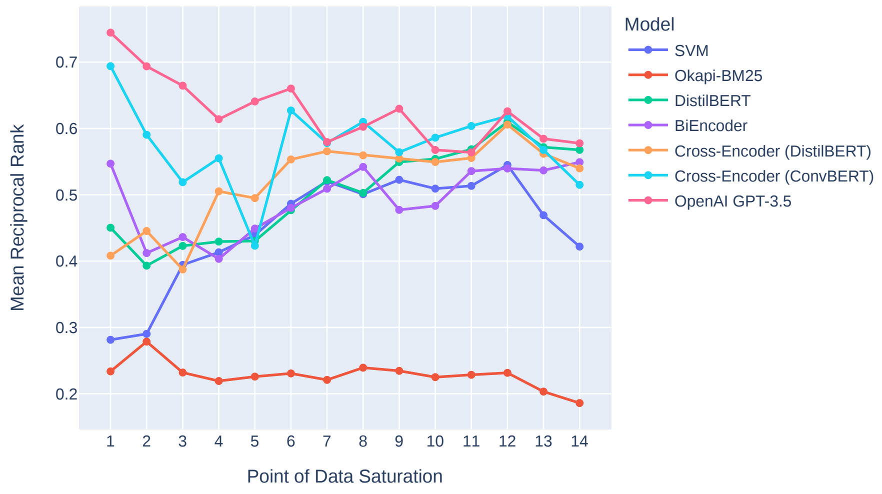


Figure A.3: Plot of MRR for coder 1 across $K = 1$ to $K = 14$ computed using original annotations.

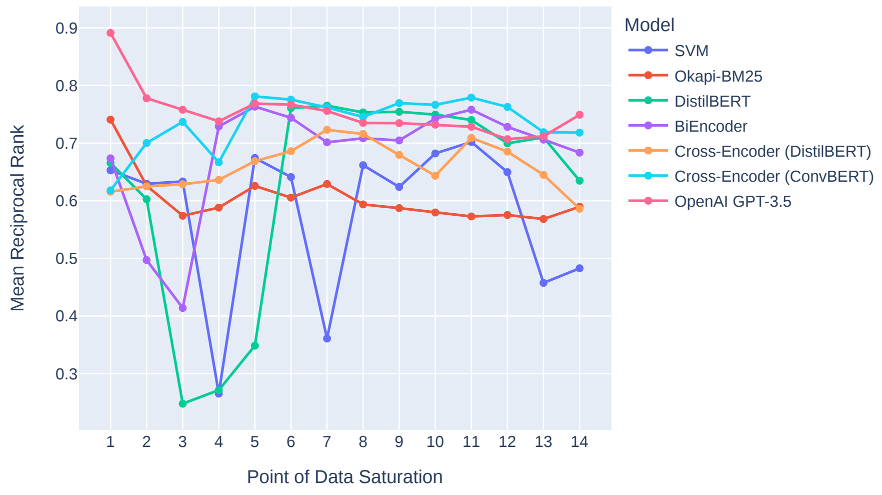


Figure A.4: Plot of MRR for coder 2 across $K = 1$ to $K = 14$ computed using original annotations.

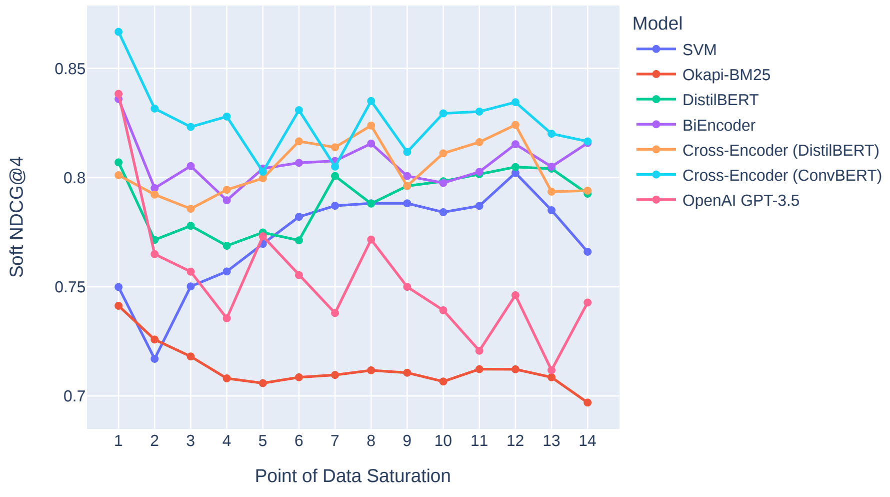


Figure A.5: Plot of sNDCG@4 for coder 1 across $K = 1$ to $K = 14$ computed using original annotations.

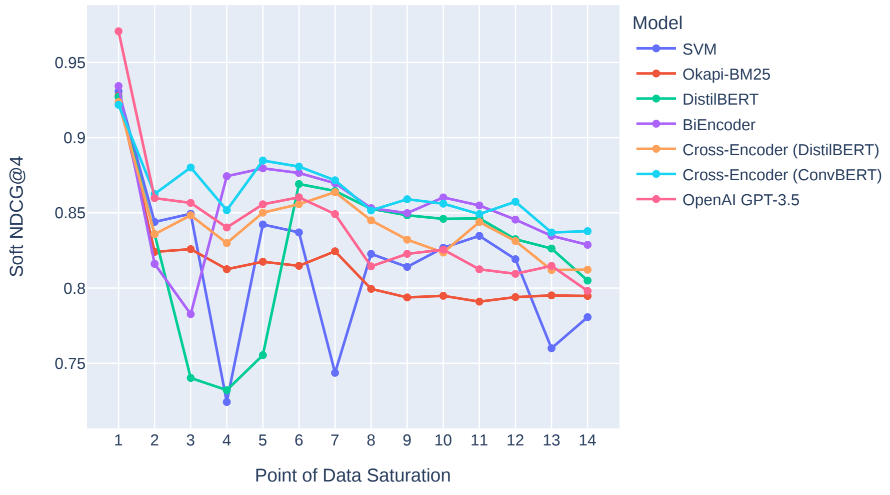


Figure A.6: Plot of sNDCG@4 for coder 2 across $K = 1$ to $K = 14$ computed using original annotations.