Institute of Information Systems and Marketing (IISM)
Kaiserstraße 89-93
76133 Karlsruhe - Germany
http://iism.kit.edu

Karlsruhe Service Research Institute (KSRI)
Kaiserstraße 89-93
76133 Karlsruhe – Germany
http://ksri.kit.edu

Towards the Design of an Interactive Machine Learning System for Qualitative Coding

Short Paper

Tim Rietz
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
tim.rietz@kit.edu

Alexander Maedche
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
alexander.maedche@kit.edu

Abstract

Coding is an important process in qualitative research. However, qualitative coding is highly time-consuming even for small datasets. To accelerate this process, qualitative coding systems increasingly utilize machine learning (ML) to automatically recommend codes. Existing literature on ML-assisted coding reveals two major issues: (1) ML model training is not well integrated into the qualitative coding process, and (2) code recommendations need to be presented in a trustworthy way. We believe that the recently introduced concept of interactive machine learning (IML) is able to address these issues. We present an ongoing design science research project to design an IML system for qualitative coding. First, we discover several issues that hinder the success of current ML-based coding systems. Drawing on results from multiple fields, we derive meta-requirements, propose design principles and an initial prototype. Thereby, we contribute with design knowledge for the intelligent augmentation of qualitative coding systems to increase coding productivity.

Keywords: Qualitative Coding, Interactive Machine Learning, Design Science Research, Design Principles

Introduction

Making sense of unstructured text in qualitative research requires researchers to annotate the text with short labels. This process is commonly known as *coding*. Qualitative coding has been described as both art and science and requires intensive training and experience (Ritchie and Lewis 2003). Researchers need to observe something notable in the data and then describe it with a code. This first-pass coding as the first step in qualitative data analysis (QDA) is an iterative process that allows researchers to produce frameworks to organize collected data and develop a nuanced understanding. Coding is time-consuming, even for small datasets, and much of the process can be painstaking and repetitive (Marathe and Toyama 2018). With large amounts of data, coding loses reliability and becomes intractable (Abbasi 2016; Chen et al. 2018).

The Era of Big Data further aggravates challenges for coding. With the increasing size of datasets created by digital communication or digital interview techniques (Rietz and Maedche 2019), manual coding techniques are limited by the available workforce (Crowston et al. 2012). For example, a recent study used a chatbot to ask open-ended questions and collected over 11,000 free-text responses, of which only about 50% could be analyzed through qualitative coding in a reasonable time (Xiao et al. 2020). Existing systems to support QDA provide only limited automation capabilities for coding. Systems such as Nvivo or INCEpTION make code recommendations using machine learning (ML). Simple approaches to making recommendations use keyword- or structure-matching to highlight sections based on user- or system-generated keywords. More sophisticated approaches use user-generated annotations to train an ML model

through supervised learning (Klie et al. 2018). However, user-centered studies suggest, that ML-based automation capabilities do not always meet user expectations (Marathe and Toyama 2018). Primarily, existing implementations fail to provide explanations for recommendations, thus lacking transparency (Chen et al. 2018). As a consequence, researchers lack trust in automated coding (Drouhard et al. 2017). Furthermore, functionality towards revising recommendations is mostly limited to accepting or rejecting a code and does not help researches with identifying flaws in coding rules. With the lack of transparent recommendations and limited capabilities for iteratively revising coding rules to train an ML-based system, qualitative researchers are reluctant to adopt ML-based support for qualitative coding (Marathe and Toyama 2018).

Still, several success stories showcase how ML techniques can accelerate certain types of QDA. For example, ML can help with identifying potentially ambiguous data to support collaborative qualitative coding (Drouhard et al. 2017) or with classifying web pages (Martens and Provost 2014). Especially the machine teaching paradigm of interactive machine learning (IML) seems promising for increasing coding productivity (Chen et al. 2018). In IML, a user iteratively builds and refines an ML model in a cycle of teaching and refinement (Dudley and Kristensson 2018). The iterative training is similar to the iterative refinement of codes and coding rules during qualitative coding. Additionally, the teaching paradigm might help researchers with building trust in recommendations, as it visualizes the effect of manual codes on automated recommendations (Marathe and Toyama 2018). While scholars across fields agree on the relevance of designing and evaluating an IML system for qualitative coding, we have yet to see studies proposing testable design principles and offering conclusive empirical results. As the first step to addressing this gap, we explore the related work to derive a set of issues, requirements and design principles for an IML system for qualitative coding, investigating the research question: *Which design principles should an interactive machine learning system for qualitative coding follow in order to establish trust in and perceived control of the system in order to increase coding productivity?*

With this short paper, we propose initial design principles and tangible design features. Furthermore, we instantiate design principles and features by showcasing a prototype IML system for qualitative coding. With our DSR project, we expect to contribute a nascent design theory for IML systems for qualitative coding that (1) helps to accelerate the coding process, (2) improves coding quality, and (3) increases the perceived trust in- and perceived control of the ML-based model.

Related Work

The Coding Process in Qualitative Data Analysis

The process of qualitative analysis transforms data into findings. There exists a wide range of approaches to the analysis process, such as ethnographic accounts, grounded theory, or content analysis. Their relevance varies based on the research domain and epistemological assumptions of a study (Ritchie and Lewis 2003). While a comprehensive review of the analysis process and individual approaches is out-of-scope for this paper, we refer to Ritchie & Lewis (2003) for in-depth information about qualitative research practices. Furthermore, Sarker, Xiao, and Beaulieu present an excellent overview of qualitative studies in information systems (Sarker et al. 2013).

Qualitative coding is a step relevant for any approach to qualitative analysis. Coding consists of devising a conceptual framework consisting of multiple codes – single words or short sentences – and applying them to the data to indicate specific information (Ritchie and Lewis 2003). Codes can constitute various levels of information and can be of *descriptive* (identifying elements and categories) or *explanatory* (developing explanations) nature. Both the development of a codebook and coding itself is an iterative, creative, human-centered process (Chen et al. 2018; Richards 2002) and essential for building a nuanced understanding of data (Marathe and Toyama 2018). However, qualitative coding is a particularly time-consuming task (Chen et al. 2016; Marathe and Toyama 2018; Xiao et al. 2020). Even for moderately sized datasets, code development and application takes hours of concentrated work, which is hard to perform reliably at scale (Crowston et al. 2012). Consequently, with access to larger datasets and advances in computer-supported analysis, the adoption of qualitative data analysis systems (QDAS) has increased substantially (Ritchie and Lewis 2003).

Qualitative Data Analysis Systems

Amongst the most prominent QDAS is Nvivo, Dedoose, and QDA Miner, to name a few[1]. These software suites cover various steps of the analysis process, like organizing documents, coding, and analysis of text and other media like audio and video. Currently, support to accelerate qualitative coding with automated procedures is limited (Marathe and Toyama 2018). Nvivo includes an experimental feature that uses machine learning to train a classifier based on user annotations. QDA Miner allows user to evaluate their datasets through automated sentiment analysis or identifying topics with topic modeling. However, these features are not designed with a focus on providing transparent suggestions, which makes them hard or sometimes impossible to validate (Grimmer and Stewart 2013). Issues with the quality of and trust in automated suggestions and a lack of integration in the process of qualitative coding have led to reluctance in adopting ML-based features (Marathe and Toyama 2018).

Furthermore, various open-source QDAS have been developed, such as INCEpTION (Klie et al. 2018), TEXTANNOTATOR, LIDA, or Aeonium (Drouhard et al. 2017). These tools are commonly web-based and modular, allowing the integration and configuration of specific use-cases. However, their predominant focus lies in supporting corpora creation tasks in the context of natural language processing (NLP). As such, developing a codebook is often cumbersome, and the integrated ML-based annotation recommendation works best for NLP-tasks, such as named entity recognition. Most importantly, these systems are not designed around building trust in recommendations through an interactive coding workflow between human and machine implementing the principles of interactive ML.

Interactive Machine Learning for Qualitative Coding

Machine learning can perform well for text classification tasks, such as for the identification of sentiment or modeling topics in unstructured text (Abbasi 2016). However, ML methods in complex contexts are at risk of lacking domain-specific user input. Interactive machine learning (IML) places the user in the center of the interaction with the ML system, aiming to create and evolve ML models iteratively through user input, thus creating a good fit to users' goals and needs (Dudley and Kristensson 2018). This approach enables users to review model outputs, adjust recommendations through feedback, and verify changes. Predominantly, IML is applied for *Interactive Labeling* tasks, in which users interact with the system to generate labels for documents, such as images or abstracts (Meza Martínez et al. 2019). Due to its human-centered approach, IML has great potential for improving the integration of automation into coding processes by providing transparent and trustworthy recommendations (Chen et al. 2016).

In the context of coding, the researcher could act as a teacher for the ML model (Knäble et al. 2019). Therein, a researcher interacts in a transparent model development process, where the model learns from iterations of code refinement through adjusting coding rules as well as accepting and rejecting recommendations (Chen et al. 2018; Crowston et al. 2012). Existing systems that provide interactive code recommendations build on the ML technique of active learning (AL) rather than IML. AL focuses on identifying new points for labeling by a user. On the other hand, IML emphasizes the role of the users during the process – the user is the driving factor for selecting points to label (Dudley and Kristensson 2018). For example, INCEpTION integrates active learning to provide annotation assistance and extends the functionalities of WebAnno (Klie et al. 2018). While Kile et al. provide an overview of use cases, no structured evaluation is provided, specifically of ML-supported coding. INCEpTION focuses on semantic annotation (attaching additional information to concepts, such as people or places) and lacks explanations for recommendations. Aeonium, an ML-based system to draw the attention of multiple coders towards potentially ambiguous data, uses ML to determine which document to label based on predicted ambiguity (Drouhard et al. 2017). Researchers in IS and HCI alike (e.g., Chen et al. 2018; Lindberg 2020; Marathe and Toyama 2018; Yan et al. 2014) have called for IML systems to assist qualitative researchers throughout the coding process. However, there seems to be no established design for an IML system for qualitative coding that is grounded in empirical evidence. Furthermore, multiple fields influence the design requirements for such a system, such as Information Systems (IS), Human-Computer Interaction (HCI), Social Science (SS), or Computer Science (CS), complicating the integration of present work. Finally, more research is needed to understand the impact of the interaction with the IML system on users' level of trust (Meza Martínez et al. 2019).

[1] For a detailed overview on software and capabilities, see e.g., De Almeida et al. 2019; Freitas et al. 2018

Methodology

The overall objective of this research project is to design an IML system that assists qualitative researchers to accelerate the coding process and improve code quality while building trust in the system. We adopt the Design Science Research (DSR) paradigm, as outlined by Hevner et al. (2004) and Kuechler and Vaishnavi (2008) to address a real-world challenge. Therefore, we iteratively create, evaluate and improve artifacts as instantiations of the design principles (DPs) outlined in the following in two cycles (Figure 1) (Hevner et al. 2004). Research cycle (1) focuses on improving the integration of IML into the coding process. To that end, we firstly identify the fundamental issues by reviewing extant literature (e.g., in IS, HCI, SS, and CS) and practical studies. Next, we derive requirements based in part on two theoretical foundations: *Toulmin's Model of Argumentation* and the *Three-Gap Framework*. Finally, we formulate preliminary design principles, and evaluate a prototype with regards to its effect on coding productivity in terms of speed and quality. In research cycle (2), we explore the effect of explanations on trust in and adoption of the IML system. In the following, we outline dominant issues of (I)ML for qualitative coding as part of the *Problem Awareness*. Subsequently, we derive requirements (R) and DPs as part of the *Suggestion* and present our prototype as well as an outlook in the *Development* section.

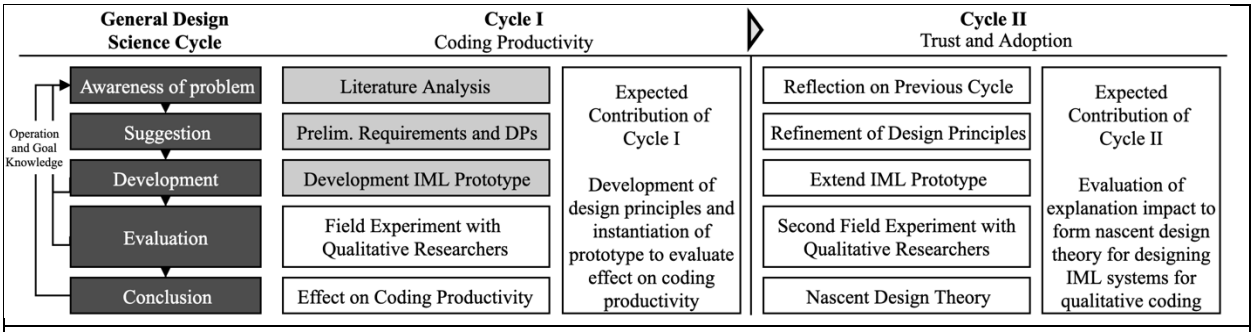

Figure 1. Research Methodology (based on Kuechler and Vaishnavi 2008)

Designing and IML system for qualitative coding

Problem Awareness

Existing literature on ML for qualitative coding suggests that two issues (I) are crucial to reliably support qualitative researchers with the coding of large datasets (Marathe and Toyama 2018):

(I1) Integration of ML model training into the process of qualitative coding. Coding does not usually create appropriate datasets for supervised or unsupervised ML model creation, as the goal of qualitative researchers commonly is not to create rich datasets for training ML models (Chen et al. 2016). In particular, while ML performs best on categories that frequently occur in the underlying data, these categories are usually not the most meaningful to researchers (Chen et al. 2018). Coding experts stated that rather than having to set up a code recommender from scratch, they prefer an interactive process of fine-tuning recommendations (Marathe and Toyama 2018). Existing implementations, such as *Nvivo's autocode*, commonly do not allow for editing and refining criteria. The integration of training into the coding process is challenging both from the perspectives of interface and algorithm design. Besides integrating revision functionalities into the interface, the algorithm has to provide high-quality recommendations even in the face of only a few annotated samples. The number of required annotations might be reduced by training a specific recommender for each code (Yan et al. 2014). On the other hand, research suggests that using human-developed coding rules rather than elaborate ML models can achieve high-quality recommendations (Crowston et al. 2012; Marathe and Toyama 2018). Investigating a combination of human- and machine-derived rules for qualitative coding seems promising. Hence, the design goal for I1 is to provide researchers with an interactive process that consists of adding codes as well as receiving and revising code recommendations.

(I2) Providing trustworthy code recommendations. Previous work on trust in recommendation agents has demonstrated the great importance of transparency for building trust (Wang and Benbasat

2016). However, systems to support coding rarely provide explanations to foster transparency, making it hard for users to predict recommendation behavior. A comparison of multiple interfaces that provide explanations for a decision-making algorithm found no influence of the interface on the user's trust in the algorithmic decision. However, interactive interfaces were able to affect users' comprehension of the algorithm (Cheng et al. 2019). There is evidence that for comprehension to induce trust, the context of the decision or recommendation made by an algorithm is crucial. While humans seem uncomfortable with algorithms making decisions that have a significant effect on human lives (such as graduate school admission), they may be less concerned about recommendations in a research context (Cheng et al. 2019). For qualitative coding, explanations can increase initial trust levels in an intelligent system, while a lack of explanations reduces trust over time (Holliday et al. 2016). Research on trust in and adoption of online recommendation agents has presented evidence that stresses the importance of a user's initial trust in a recommender for adoption behavior (Wang and Benbasat 2005). Therefore, it is crucial to support the understanding of recommendations as a way of building trust and foster the adoption of an IML system. More research is required on designing explanations that build trust in the algorithm while guiding users when revising coding rules. Hence, the design goal for I2 is to provide researchers with explanations for recommendations to build trust in the system.

Suggestion

The first design goal (I1) centers around the interaction between human and machine. It aims to provide researchers with a coherent process for coding a dataset with machine support to increase coding productivity by increasing speed and quality. Therefore, both human and machine must agree on the characteristics of a code in the data. An important characteristic is the unit of analysis, defining the level at which annotations are made to the text (e.g., can annotations be made freely or are they locked to word, sentence, or paragraph level). While the unit of analysis can have a significant effect on computing intercoder reliability, special-purpose QDA software usually does not allow researchers to explicitly define a unit of analysis for a project (Marathe and Toyama 2018). A flexible unit of analysis makes it difficult to exactly match the boundaries of text to be captured by machine recommendations (Crowston et al. 2012). Therefore, the unit of analysis should be fixed for a given document **(R1)**. Furthermore, human and machine need to follow a common goal – structuring a dataset through accurate and insightful codes. As part of the interaction of the two parties in the IML paradigm, the human is responsible for the accuracy and detail of codes (as the domain expert). At the same time, the machine assists in increasing speed or improving accuracy and detail. Researchers should not have to concern themselves with ML-related tasks perceived as additional work in the coding process, without gaining additional benefit. Comparing recommendations against the own coding intuition for every recommendation is perceived as adding additional steps to coding, increasing interaction costs (Marathe and Toyama 2018). As such, training the ML model needs to be integrated into the coding process seamlessly **(R2)**. Therefore, an initial manual coding should inform code recommendations. Researchers value the process of familiarizing and coding parts of their data to get a better understanding and identify the direction of the analysis. However, they desire recommendations to reduce repetitiveness once saturation of the codebook has been achieved (Marathe and Toyama 2018). Recommendations need to incorporate manual annotations and gradually support the researcher as the codebook becomes more refined **(R3)**. A completed codebook should contain precise rules that outline how to code data. These rules can conceal uncertainty on the part of the researcher as they reduce the complexity of the underlying information. Sometimes codes are not assigned with high confidence but because no other codes fit. As such, codes require some sort of keywords, definitions, or rules to be interpretable (Ganji et al. 2018). Rule-based code recommendations show promising results for qualitative coding, but can be time-intensive to define (Crowston et al. 2012; Marathe and Toyama 2018). Furthermore, defining coding rules can help researchers better to understand data (Grimmer and Stewart 2013). An IML interface should assist researchers in defining coding rules as part of developing a codebook **(R4)**. Adding to R2, recommendations can provide benefits additional to accelerating coding, which is identifying coding errors. Coding rules and definitions usually are not final upon definition and are iteratively revised as additional data is coded (Richards 2002). Imprecise codes usually become apparent as data is re-coded by a second coder. In alignment with R2, code recommendations might act as a proxy for the second coder in evaluating coding rules and coding rigor (Chen et al. 2016; Marathe and Toyama 2018). The IML interface should enable researchers to identify potential issues and reflect on coding rules to provide an added benefit to utilizing ML support, particularly for experts to qualitative coding **(R5)**. Finally, the interaction between human and machine should follow existing design principles for IML

interfaces (Dudley and Kristensson 2018). In the context of qualitative coding, it is particularly important to support the understanding of model uncertainty and confidence to encourage coders to approach the coding process as part of theory-building, particularly novices (Richards 2002). Goals and tasks should be made explicit **(R6)**. Aggregating R1–R6, we propose:

> **DP1**: *Provide the IML system with a coding process that assists with defining and revising code rules based on a predefined unit of analysis, gradually supporting researchers with code recommendations that accelerate coding and identify errors.*

The first design goal focuses on increasing coding productivity and assumes that a researcher trusts the system to provide useful recommendations. To that end, the second design goal (I2) centers around how to design transparent recommendations by providing explanations to build trust. Research on trust in recommendation agents has demonstrated the relevance of trust for the intention to adopt an agent (Wang and Benbasat 2005). As such, it is vital to provide recommendations that can build trust for the system to be adopted by practitioners for increasing coding productivity. The ability to explain the recommendations of an algorithm is related to the complexity of the algorithm. Complex ML models tend to become black-boxes. Interestingly, complex ML models do not necessarily outperform simple approaches. A comparison of human-developed with ML-generated coding rules showed that for straightforward codes (such as identifying an apology), simple rules performed best (Crowston et al. 2010). When developing ML models, the number of input factors does not correlate with the predictive accuracy of the model (Yan et al. 2014). As such, the IML system should utilize simple ML models to provide recommendations **(R7)**. Furthermore, researchers need to be able to relate recommendations to theoretical foundations, e.g., underlying coding rules or dictionaries (Chen et al. 2018). Researchers not trained in ML techniques may otherwise be tempted to reject recommendations altogether or accept them to quickly in the case of novices. Additionally, it is essential to show which features (e.g., keywords) anchor a recommendation to enable researchers to predict *how* changes affect recommendations (Cheng et al. 2019). Counterfactual explanations that determine which features of an annotation (such as specific words) could be slightly different to change the recommended code are promising for increasing understanding (Martens and Provost 2014). However, explanations commonly cater to ML experts or users with high technical literacy (Cheng et al. 2019). Code recommendations should be understandable and predictable for researchers independent of their level of technical expertise. Based on *Toulmin's Model of Argumentation,* a rule-based explanation should include a data premise (the annotation), certainty factor (certainty of the algorithm), and a conclusion (the resulting code) to build trust and acceptance (Gregor and Benbasat 1999). Overall, recommendations should include rule-based and counterfactual explanations, as well as a certainty factor **(R8)**. Following R5, *Cognitive Learning Theory* suggests that expert coders can use explanations to verify manual coding and resolve anomalies (disagreements between own perception and recommendation) (Gregor and Benbasat 1999). Aggregating R4, R5, R7 & R8, we propose:

> **DP2**: *Provide the IML system with rule-based and counterfactual explanations for code recommendations that make the ML model transparent to enable researchers to understand the algorithm and predict recommendation behavior.*

Development

We propose four design features (DFs) to instantiate the outlined design principles in an initial prototype. Following DP1, the IML system combines revisable hand-coded rules with supervised ML to make code recommendations that accelerate coding (**DF1**). For code recommendations, we display a confidence value, and the user can accept or reject recommendations without diving into explanations. Furthermore, the system highlights (miss)matches between manual annotations and predicated codes to enable researchers to identify coding errors (**DF2**). Therefore, the system highlights the conflict with a special icon (lightning) alongside a confidence value and the predicted label. Following DP2, the IML system explains code recommendations by referencing underlying code rules and providing a certainty factor (**DF3**). The user can access explanations through the coding context menu that opens every time a user makes a new annotation or clicks on an existing one. Keywords that are included in the currently relevant paragraph (the unit of analysis in the example) are highlighted in the coding rule. We adopt a query-style format for defining coding rules (Marathe and Toyama 2018). The rule can be revised directly in the context menu. On top, we provide counterfactual explanations to increase researchers' understanding and prediction capabilities for recommendations (**DF4**). We build on the foundations laid by Martens and Provost (2014)

based on the *Three-Gap Framework* to provide counterfactual explanations. The framework aligns three different models to consider when providing explanations to improve decision making: the user's model, the system's model, and reality. As such, we explain code recommendations by showing researchers the (minimal) set of words in an annotation that is responsible for a given classification (Martens and Provost 2014). Thus, users may identify relevant keywords that might be missing from the coding rule. The calculation of counterfactual explanation is similar to the common notion of *Shapley Values*, which explain a prediction by highlighting the impact of individual features. The impact of features (each word in a sentence in the case of Cody) is calculated by predicting a label while removing one (or multiple) words from a sentence. Other common visualizations to explain ML predictions, such as *Partial Dependence Plots* or *Individual Conditional Expectations,* could be considered if users desired deeper insights into how specific words trigger predictions for a label. Figure 2 showcases the interface of our IML system prototype and highlights DFs.

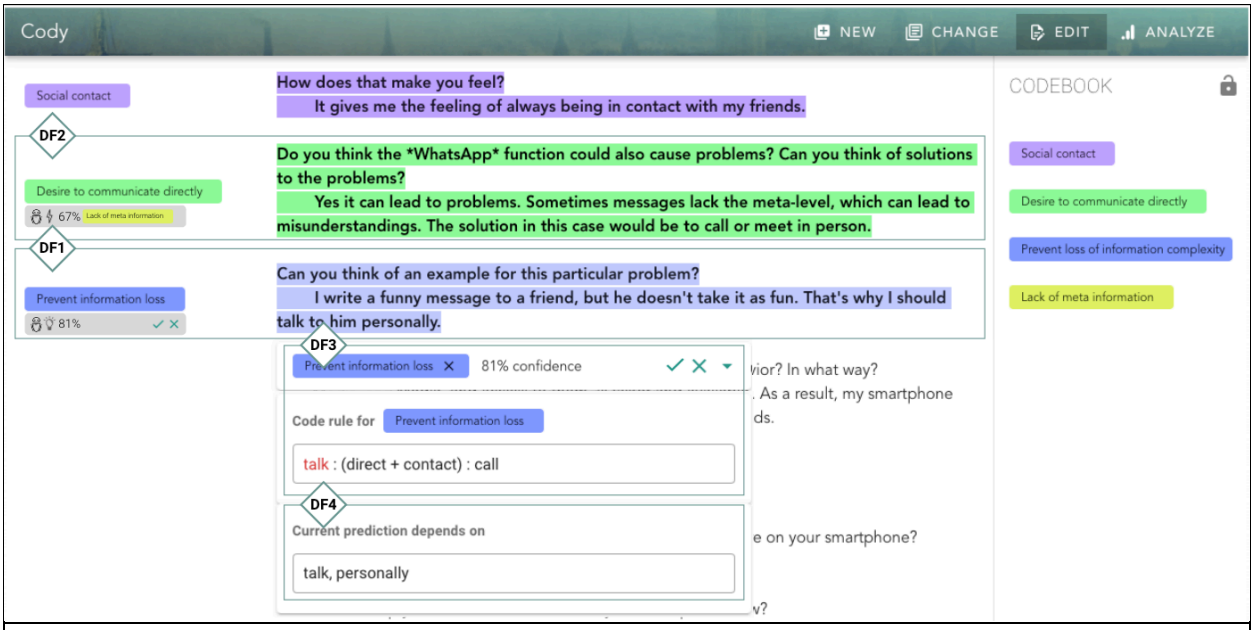

Figure 2. Prototype of IML system Cody with highlighted design features.

Currently, we are developing the IML system based on DF1-4 using Vue as a front end framework and Flask to connect the front end to a Python back end. The code recommender uses a combination of query-matching (to identify codes based on hand-coded rules, using the Python library *whoosh*) and Logistic Regression (fitted with a Stochastic gradient Descent approach, using the Python library *scikit-learn*) to make code recommendations. Furthermore, the IML system suggests a default code rule to the user upon creating a new label. This suggestion is made by comparing the label with the respective annotation using a precompiled language model from the *spaCy* library for natural language processing.

The proposed code recommender faces multiple challenges during implementation. For labels that are highly context-depending, users might find it difficult to define rules that consider implicit knowledge, limiting the utility of query-matching. These instances stress the importance of aiming for IML systems to augment users in qualitative coding, rather than developing systems that strive to make "perfect" suggestions. In the end, coding is a subjective process depending on the context, research discipline, and epistemological assumptions that rarely has a definitive correct answer. As such, the proposed IML system uses code rules and ML model exclusively for each document and user. The predictive quality of a supervised ML model at the same time is limited by the available amount of labeled training data. As users should not be expected to provide sufficient data for model training (I1), we need to devise strategies for making the most of the limited data available. The IML system uses available query-matching suggestions for ML model training. Additionally, the system considers sections that users skipped during coding *negative examples* for the training of the ML model. For more insights on the technical implementation of the IML system, please refer to Rietz et al. 2020. We a currently conducting both a formative (focusing on

the coding interface) and summative (comparison with state-of-the-art QDAS) evaluation of the IML prototype with qualitative researchers and real data to understand the effect of our design features and demonstrate the feasibility of the system for different types of text corpora.

Conclusion and Next Steps

This paper presents first insights from an ongoing DSR project on designing an IML system for qualitative coding to assist qualitative researchers with coding large datasets. We discuss issues of systems for qualitative coding, derive requirements, and propose preliminary design principles. Currently, we are instantiating a prototype to evaluate it in a field experiment with qualitative researchers. We expect to contribute a nascent design theory (Gregor and Hevner 2013) for the class of IML interface artifacts in the context of qualitative coding. Therein, we provide an *Improvement* type contribution according to the DSR contribution framework by Gregor and Hevner (2013) as we develop and evaluate a new solution for a known problem. In evaluating the outlined design principles, we explore a possible design to enable a continuous and iterative interaction between human and machine. Furthermore, we evaluate different forms of explanations in a field setting. While we focus on IML for qualitative coding, we assume that our findings could also be valuable for increasing trust in IML systems in organizational settings.

While our research follows established guidelines for conducting DSR (Kuechler and Vaishnavi 2008), some limitations remain. Due to the early stage of the research project, the outlined requirements and DPs are preliminary and illustrate the intended design of the ILM system. *Toulmin's Model of Argumentation* and the *Three-Gap Framework* inform especially DP2 and its DFs. Different theoretical foundations could have been selected to guide the design, which might have resulted in different DPs. Furthermore, we recognize the essential role of perceived system ability for determining trust in a system (Holliday et al. 2016). As such, it will be crucial for an IML system to provide high-quality recommendations. To that end, we utilize a combination of rule-based recommendations and supervised ML. This approach has not yet been evaluated in a system for coding but combines two techniques that individually showed promising results. Our design focuses on the interaction between a human coder and the recommender, and does not consider other relevant aspects of coding, such as collaboration between human coders. The integration of multiple human coders constitutes an exciting challenge for future research.

References

Abbasi, A. 2016. "Big Data Research in Information Systems: Toward an Inclusive Research Agenda," *Journal of the Association for Information Systems* (17:2), pp. 1–32.

De Almeida, C. A., Freitas, F., Costa, A. P., and Moreira, A. 2019. "WEBQDA: The Quest for a Place in the Competitive World of CAQDAS," in *Proceedings of the 2019 International Conference on Engineering Applications (ICEA)*.

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 Transactions on Interactive Intelligent Systems* (8:2), pp. 1–21.

Chen, N., Kocielnik, R., Drouhard, M., Suh, J., Cen, K., Zheng, X., Aragon, C. R., and Pena-Araya, V. 2016. "Challenges of Applying Machine Learning to Qualitative Coding," in *ACM SIGCHI Workshop on Human-Centered Machine Learning*.

Cheng, H. F., Wang, R., Zhang, Z., O'Connell, F., Gray, T., Harper, F. M., and Zhu, H. 2019. "Explaining Decision-Making Algorithms through UI: Strategies to Help Non-Expert Stakeholders," *Proceedings of the Conference on Human Factors in Computing Systems (CHI '19)*, pp. 1–12.

Crowston, K., Allen, E. E., and Heckman, R. 2012. "Using Natural Language Processing Technology for Qualitative Data Analysis," *International Journal of Social Research Methodology* (15:6), pp. 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), pp. 1–2.

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," *IEEE Pacific Visualization Symposium*, pp. 220–229.

Dudley, J. J., and Kristensson, P. O. 2018. "A Review of User Interface Design for Interactive Machine

Learning," *ACM Transactions on Interactive Intelligent Systems* (8:2).
Freitas, F., Ribeiro, J., Brandão, C., de Souza, F. N., Costa, A. P., and Reis, L. P. 2018. "In Case of Doubt See the Manual: A Comparative Analysis of (Self)Learning Packages Qualitative Research Software," *Advances in Intelligent Systems and Computing* (621), pp. 176–192.
Ganji, A., Orand, M., and McDonald, D. W. 2018. "Ease on down the Code: Complex Collaborative Qualitative Coding Simplified with 'Code Wizard,'" *Proceedings of the ACM on Human-Computer Interaction*.
Gregor, S., and Benbasat, I. 1999. "Explanations from Intelligent Systems: Theoretical Foundations and Implications for Practice," *MIS Quarterly: Management Information Systems* (23:4), pp. 497–530.
Gregor, S., and Hevner, A. R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly* (37:2), pp. 337–355.
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), pp. 267–297.
Hevner, A., March, S., Park, J., and Ram, S. 2004. "Design Science Research in Information Systems," *MIS Quarterly* (28:1), pp. 75–105.
Holliday, D., Wilson, S., and Stumpf, S. 2016. "User Trust in Intelligent Systems: A Journey over Time," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI '16)*, pp. 164–168.
Klie, J.-C., Bugert, M., Boullosa, B., de Castilho, R. E., and Gurevych, I. 2018. "The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation," *Proceedings of the International Conference on Computational Linguistics*, pp. 5–9.
Knäble, M., Nadj, M., and Maedche, A. 2019. "Oracle or Teacher ? A Systematic Overview of Research on Interactive Labeling for Machine Learning," in *Internationaler Kongress Für Wirtschaftsinformatik 2020*.
Kuechler, B., and Vaishnavi, V. 2008. "Theory Development in Design Science Research: Anatomy of a Research Project," *European Journal of Information Systems*, pp. 489–504.
Lindberg, A. 2020. "Developing Theory through Integrating Human and Machine Pattern Recognition," *Journal of the Association for Information Systems* (21:1), pp. 90–116.
Marathe, M., and Toyama, K. 2018. "Semi-Automated Coding for Qualitative Research: A User-Centered Inquiry and Initial Prototypes," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI'18)*, pp. 1–12.
Martens, D., and Provost, F. 2014. "Explaining Data-Driven Document Classifications," *MIS Quarterly* (38:1), pp. 73–99.
Meza Martínez, M. A., Nadj, M., and Maedche, A. 2019. "Towards an Integrative Theoretical Framework of Interactive Machine Learning Systems," *Proceedings of the 27th European Conference on Information Systems (ECIS2019)* (2014), pp. 1–19.
Richards, L. 2002. "Qualitative Computing—a Methods Revolution?," *International Journal of Social Research Methodology* (5:3), pp. 263–276.
Rietz, T., and Maedche, A. 2019. "LadderBot: A Requirements Self-Elicitation System," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, IEEE, September, pp. 357–362.
Rietz, T., Toreini, P., and Maedche, A. 2020. "Cody : An Interactive Machine Learning System for Qualitative Coding," in *Adjunct Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2020)*.
Ritchie, J., and Lewis, J. 2003. "Qualitative Research Practice: A Guide for Social Science Students and Researchers," *SAGE Publications* (Vol. 1).
Sarker, S., Xiao, X., and Beaulieu, T. 2013. "Qualitative Studies in Information Systems: A Critical Review and Some Guiding Principles," in *MIS Quarterly* (Vol. 37), pp. 3–18.
Wang, W., and Benbasat, I. 2005. "Trust in and Adoption of Online Recommendation Agents," *Journal of the Association for Information Systems* (6:3), pp. 72–101.
Wang, W., and Benbasat, I. 2016. "Empirical Assessment of Alternative Designs for Enhancing Different Types of Trusting Beliefs in Online Recommendation Agents," *Journal of Management Information Systems* (33:3), pp. 744–775.
Xiao, Z., Zhou, M. X., Liao, Q. V., Mark, G., Chi, C., Chen, W., and Yang, H. 2020. "Tell Me About Yourself: Using an AI-Powered Chatbot to Conduct Conversational Surveys," *ACM Transactions on Computer-Human Interaction* (1:1).
Yan, J. L. S., McCracken, N., Zhou, S., and Crowston, K. 2014. "Optimizing Features in Active Machine Learning for Complex Qualitative Content Analysis," in *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, pp. 44–48.