Cody: An Interactive Machine Learning System for Qualitative Coding

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

Peyman Toreini
Karlsruhe Institute of Technology (KIT)
peyman.toreini@kit.edu

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

ABSTRACT

Qualitative coding, the process of assigning labels to text as part of qualitative analysis, is time-consuming and repetitive, especially for large datasets. While available QDAS sometimes allows the semi-automated extension of annotations to unseen data, recent user studies revealed critical issues. In particular, the integration of automated code suggestions into the coding process is not transparent and interactive. In this work, we present *"Cody"*, a system for semi-automated qualitative coding that suggests codes based on human-defined coding rules and supervised machine learning (ML). Suggestions and rules can be revised iteratively by users in a lean interface that provides explanations for code suggestions. In a preliminary evaluation, 42% of all documents could be coded automatically based on code rules. Cody is the first coding system to allow users to define query-style code rules in combination with supervised ML. Thereby, users can extend manual annotations to unseen data to improve coding speed and quality.

Author Keywords

Qualitative coding; Supervised machine learning

CCS Concepts

•**Human-centered computing** → *Graphical user interfaces;* **Human computer interaction (HCI);**

INTRODUCTION

Annotating interview transcripts with descriptive or inferential labels, commonly known as qualitative coding, is an essential step in analyzing qualitative data to assist concept or theory development [3]. Unfortunately, coding natural language corpora is a painstaking process due to being time-intensive and repetitive [14, 8]. With access to larger datasets due to new possibilities for scalable data collection [11, 13], coding loses reliability and becomes intractable [1, 2].

Researchers use various QDAS to simplify qualitative coding (e.g., MaxQDA) [7]. Some systems incorporate machine learning (ML) for making code suggestions based on human annotations. However, recent user studies demonstrated two critical shortcomings that impede their utility for enabling qualitative coding at a scale [5, 8]: (i) ML is not integrated into QDAS in the form of an interactional process that involves refining automated suggestions. Interaction between the user and the ML model is mostly restricted to accepting and rejecting codes without insight into underlying coding rules; (ii) Therefore, code suggestions lack transparency, causing a reluctancy on the part of qualitative researchers to adopt ML-based support for qualitative coding.

In this paper, we introduce *"Cody"*, a system to increase the speed and quality of qualitative coding through interactive machine learning (IML). In IML, a user iteratively builds and improves a ML model in a cycle of teaching and refinement [6]. While users work through transcripts with Cody, the system supports users in defining and interactively revising code rules. Cody uses code rules to perform search-style query matching to automate coding partially. Furthermore, Cody learns over time by continually re-training a supervised ML model on available codes, to provide additional code suggestions once the model passes several quality thresholds. With Cody we extend previous studies [4, 8] by designing and developing a user-facing interface for semi-automated qualitative coding utilizing code rules and ML.

CODY

Cody is a system that emphasizes human-in-the-loop for qualitative coding. Users can specify the desired unity-of-analysis, add annotations and labels, define coding rules, react to suggestions, and access a rudimentary statistics page. Figure 1 shows the interface of Cody during the coding process.

Cody Interface: Cody's interface is made up of three primary sections (Figure 1): the annotation text area in the middle, the label overview on the left, and the codebook on the right. In the *annotation text area*, the user can highlight text to add a new annotation. Text highlights are automatically adjusted based on the chosen unit-of-analysis. Using the label menu, users can create new codes, assign them to annotations, or make changes. More importantly, the user can define a search-query style code rule for each label. When creating a new label, Cody suggests a code rule by comparing the chosen label with the current annotation using natural language processing and Levenshtein distance. For annotations that Cody suggests to the user, an explanation is shown. The explanation depends on the source of the suggestion - code rule or ML model. The *label overview* displays the label of each annotation, alongside

UIST '20 Adjunct, October 20–23, 2020, Virtual Event, USA

ISBN 978-1-4503-7515-3/20/10.

DOI: 10.1145/3379350.3416195

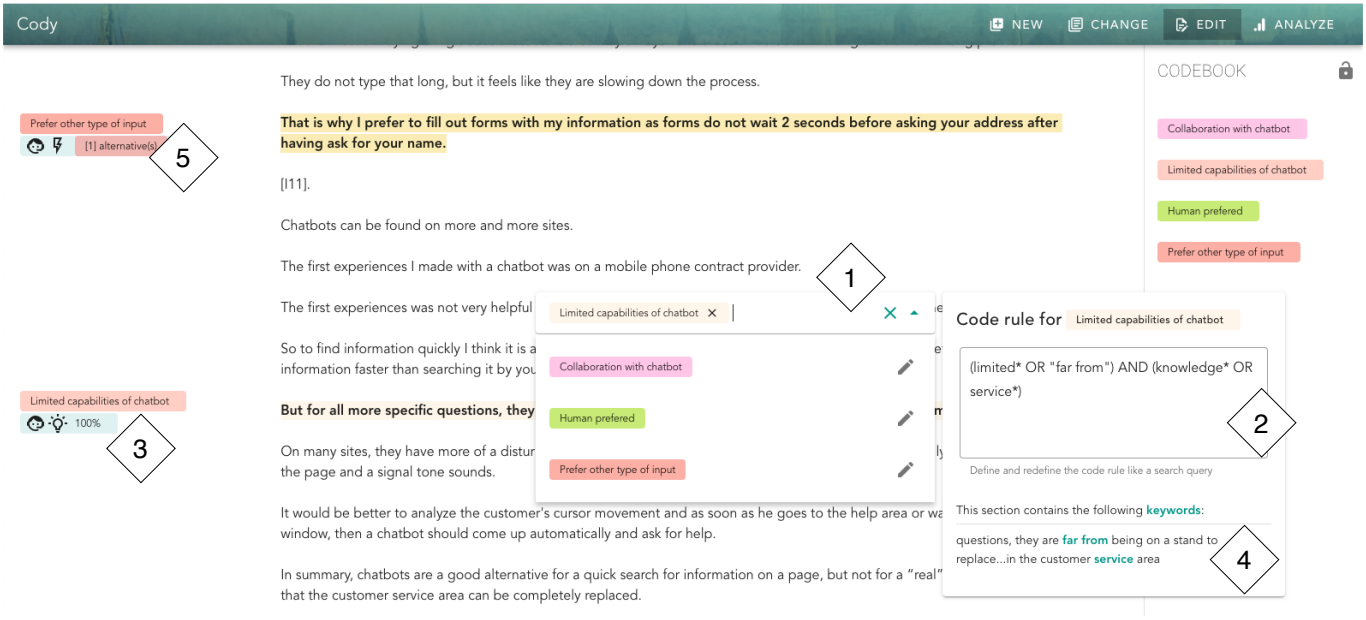

Figure 1. Screenshot of Cody edit view. (1) Users can add one or multiple labels to new or existing annotations. (2) For each label, users can define a query-style code rule. (3) Based on code rules and the underlying ML model, Cody suggests annotations. (4) Cody provides explanations for suggestions based on critical words. (5) Conflicts between multiple code rules or between rule-based and ML-based suggestions are highlighted.

an indicator for automated code suggestions. It also highlights those instances were multiple codes have been suggested. The *codebook* shows every label in use, in an order that can be defined by the user.

Code Suggestions: Cody uses two strategies to make code suggestions - code rules and supervised ML. Suggestions based on *code rules* can be provided when a rule is defined. However, a certain amount of labeled data is required to make ML-based suggestions. Cody utilizes both manual annotations and rule-based suggestions to kick-start model training. As *supervised ML* algorithm, Cody trains a logistic regression with stochastic gradient descent learning to classify unseen data based on the available annotations (positive examples). In the multiclass case, we usually face with a low number of positives for each label but lack explicit negative examples (annotations indicating the not-presence of a label). Assuming that the user makes annotations from top to bottom, Cody treats unlabeled sections of text above the last manual annotation as negatives to improve model accuracy. Furthermore, we draw from the S-EM algorithm for PU learning to reduce the number of inaccurate suggestions [12]. As such, we sample S spies from the labeled training data, so that $|S| = 0.1 \times |L|$. For every spy *(s)*, we analyze the accuracy of the model's prediction. Cody will only display label suggestions to the user for label *(l)* from all labels *(C)* that were correctly suggested for all spies, thereby prioritizing precision over recall, i.e.,

$$l = \{\, l \in C : \forall s \in S : q(s|l) = s|l \,\}$$

Explaining suggestions: Cody highlights critical keywords for a suggestion in the label menu, which is opened by clicking on a suggestion. For rule-based suggestions, matched words are highlighted in an excerpt from the current annotation. For ML-based annotations, Cody displays the indicative words for a suggestion. If these indicative words were to change, the ML would make a different suggestion for the annotation (heuristic approach, c.f. [9]).

PRELIMINARY EVALUATION

During the preparation for an upcoming in-depth evaluation, we used Cody for the qualitative coding of a dataset of laddering interviews. Two coders coded the same 807 paragraphs, one using Cody and the other MAXQDA. While the coder with MAXQDA made every annotation manually, the coder with Cody used rule-based annotation suggestions. ML-based suggestions were not used in this initial pretest.

Overall, with Cody, only 58% of the dataset had to be coded manually, with the remaining 42% being automated suggestions. Two independent researchers created a gold-standard for the dataset with standard procedure [10]. Compared to the gold-standard codes, the manual annotations were correct in 77% of all cases. In comparison, automated suggestions were correct in 68% of all cases. Intercoder agreement between the MAXQDA coder and the Cody-coder was 72%.

CONCLUSION & FUTURE WORK

In this paper, we introduce Cody, an interactive ML system for qualitative coding that provides code suggestions via code rules and supervised ML. We outlined the components of Cody, which include a lean coding interface for users to make annotations and iterate code rules, while receiving explainable code suggestions. In a preliminary evaluation, we demonstrated that 42% of all codes could be automatically generated using only code rules, with 68% of the suggested codes being correct. Our next step is to evaluate Cody with qualitative researchers who experiment with the tool on their own data.

REFERENCES

[1] Ahmed Abbasi, Suprateek Sarker, and Roger HL Chiang. 2016. Big data research in information systems: Toward an inclusive research agenda. *Journal of the Association for Information Systems* 17, 2 (2016), 3.

[2] Nan-Chen Chen, Margaret Drouhard, Rafal Kocielnik, Jina Suh, and Cecilia R Aragon. 2018. Using machine learning to support qualitative coding in social science: Shifting the focus to ambiguity. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8, 2 (2018), 1–20.

[3] Nan-chen Chen, Rafal Kocielnik, Margaret Drouhard, Vanessa Peña-Araya, Jina Suh, Keting Cen, Xiangyi Zheng, and Cecilia R Aragon. 2016. Challenges of applying machine learning to qualitative coding. In *ACM SIGCHI Workshop on Human-Centered Machine Learning. http://hcml2016. goldsmithsdigital. com/program*.

[4] Kevin Crowston, Eileen E Allen, and Robert Heckman. 2012. Using natural language processing technology for qualitative data analysis. *International Journal of Social Research Methodology* 15, 6 (2012), 523–543.

[5] Margaret Drouhard, Nan-Chen Chen, Jina Suh, Rafal Kocielnik, Vanessa Pena-Araya, Keting Cen, Xiangyi Zheng, and Cecilia R Aragon. 2017. Aeonium: Visual analytics to support collaborative qualitative coding. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 220–229.

[6] John J Dudley and Per Ola Kristensson. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8, 2 (2018), 1–37.

[7] Fábio Freitas, Jaime Ribeiro, Catarina Brandão, Francislê Neri de Souza, António Pedro Costa, and Luís Paulo Reis. 2017. In case of doubt see the manual: a comparative analysis of (self) learning packages qualitative research software. In *International Symposium on Qualitative Research*. Springer, 176–192.

[8] Megh Marathe and Kentaro Toyama. 2018. Semi-automated coding for qualitative research: A user-centered inquiry and initial prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[9] David Martens and Foster Provost. 2014. Explaining data-driven document classifications. *Mis Quarterly* 38, 1 (2014), 73–100.

[10] Michael Quinn Patton. 2014. *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications.

[11] Tim Rietz and Alexander Maedche. 2019. LadderBot: A requirements self-elicitation system. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 357–362.

[12] Stefan Schrunner, Bernhard C Geiger, Anja Zernig, and Roman Kern. 2020. A generative semi-supervised classifier for datasets with unknown classes. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 1066–1074.

[13] Ella Tallyn, Hector Fried, Rory Gianni, Amy Isard, and Chris Speed. 2018. The Ethnobot: Gathering Ethnographies in the Age of IoT. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

[14] Jasy Liew Suet Yan, Nancy McCracken, and Kevin Crowston. 2014. Semi-automatic content analysis of qualitative data. *iConference 2014 Proceedings* (2014).