Structuring Human-ML Interaction With an Immersive Interface Based on Qualitative Coding

Johanne Christensen[*]
North Carolina State University

Ben Watson[†]
North Carolina State University

ABSTRACT

With ever increasing bodies of data, much of it unlabeled and from complex, dynamic and weakly structured domains, machine learning (ML) is more necessary than ever. Yet even domain experts have difficulty understanding most ML algorithms, and so cannot easily retrain them as new data arrives. This limits ML's use in many fields that sorely need it, such as law, where users must have confidence in ML results. Interactive machine learning techniques have been proposed to take advantage of humanity's ability to categorize in these complex domains, but little attention has been paid to building interfaces for non-ML experts to provide input, and in particular to creating a user experience that engenders trust. Qualitative coding - the decades-old practice of manual classification - provides a proven methodology that can be adapted to structure interaction between domain experts and ML algorithms. Qualitative coders often use physical props such as notecards to help sort through and understand datasets. Here we explore how an immersive system can be built to leverage QC's intuitive techniques and grow a trusting partnership between human and ML classifiers.

Index Terms: H.5.2 [INFORMATION INTERFACES AND PRESENTATION]: User Interfaces—User-centered design; I.5.5 [Pattern Recognition]: Implementation—Interactive Systems

1 INTRODUCTION

As the amount of data we generate and collect continually increases, the need for computer-aided analysis grows. Although machine learning (ML) algorithms exist for such tasks as classifying data, two issues have arisen as these algorithms transitioned from lab study to real world use.

Firstly, for end users, tuning a robust ML model generally needs expert assistance, since such tuning normally requires understanding the ML algorithm's inner workings. Without such understanding, user confidence in the accuracy of the generated models tends to be low, which has hampered the adoption of these techniques, particularly in domains such as law, where users have culpability if results are incomplete, and must convince skeptical courts of their correctness.

Secondly, while machine learners perform well in clearly defined domains, there are still many complex domains where available data sets do not fully enable ML because they may be incomplete or uncertain. Human domain experts can usually classify small data sets in these domains, but the resulting data scale does not support effective ML.

'Human-in-the-loop' systems such as interactive machine learning [7] leverage the capabilities of both human and machine by fostering a dialogue between them centered around model creation and refinement. Such a system empowers end users to tune ML models, increases their confidence in the model's results, and leverages human expertise to improve the algorithm's results. Research on these systems has focused on accommodating such diaglogue in ML algorithms, rather than interfaces that might support the dialogue itself.

Related work on visualization for ML has recently begun [13]. However, this work is meant to aid ML experts in their work, rather than to help domain experts. The use of large displays for visual analytics in large datasets has been proprosed to better support sensemaking via interactive visual exploration and a more natural interface than traditional interface elements [4] [5] [3] [2]. Software such as iCluster [1] utilizes techniques drawn from analog methods of clustering documents. While the above techniques focus on providing easier interaction for domain experts, as yet, we have not seen any work that focuses on classification rather than clustering.

To address this lack, we propose structuring this dialogue with the methods of qualitative coding (QC), a decades-old process of manual analysis. Widely used in the social sciences, QC categorizes data to facilitate further study. Manual classification can be overwhelming, and in practice, qualitative coders often utilize their environment to build 'data displays' essentially, an analog immersive environment, with notecards on a table or post-its on a board that can be moved around to sort and categorize data (see figure 1).

If a system utilizing QC techniques to improve ML classification is to succeed, an immersive platform will be an essential tool for providing this more natural technique of interacting with large sets of data. In this paper, we explore how such a platform could be implemented.

2 QUALITATIVE CODING

Qualitative coding (QC) [11] is a manual classification method commonly employed in the humanities and behavioral sciences used to extract meaning from non-numeric data such as text, imagery, and video. Users, called 'coders', work independently in an iterative process to review data, label it with codes, group codes into categories, draw themes from categories, and finally build theory from themes (Fig 2). The process of labelling data with codes is called first cycle coding. Here researchers perform high-volume work, making few high level inferences. The remainder of the process is second cycle coding, during which researchers generate theoretical conclusions based on the overall data patterns found in the first cycle.

It is possible for one coder to use this method to process data, however it is also common for multiple coders to work on the same data set, which allows for verification of each other's work. When multiple coders collaborate, a measure of intercoder agreement, such as Cohen's kappa or Krippendoff's alpha, is used to evaluate the robustness of the labeled data. If agreement is not high enough, iteration of the first cycle of coding continues.

Using a form of QC called grounded theory, labels are often created as needed by coders, rather than working with a set of labels defined rigidly before beginning. This allows theories to develop organically from the data. In order ensure the validity, reliability and sensitivity of its results, QC implements several tools and practices to assist coders in maintaining a distinct set of labels.

The *codebook* is an index of labels used in the first cycle, which includes a definition, inclusion and exclusion criteria, typical examples, and boundary cases. Each coder also maintains *analytic memos*

[*]e-mail: jtchrist@ncsu.edu
[†]e-mail: bwatson@ncsu.edu

Figure 1: Analog methods of categorizing data.

which they attach to the data explaining their reasoning for their code choice, any concerns about their choice, or any other notes they may wish to communicate to other coders (or themselves if they are working alone) that do not fit into the codebook. *Data displays* are a blanket term referring to any method of organization that coders use to visually summarize their work. This can be anything from an array of post-its to organizing data into piles of hard copies on tables. Finally, as manual coding cannot be easily undone, it is common to keep a *history log*.

Although many QC researchers still prefer analog methods, computer-assisted qualitative data analysis software (CAQDAS) exists, and many make use of ML to partially automate coding. They also include features to track codebooks, history, memos and make visualizations replacing data displays.

3 Interactive Systems for QC/ML Interactions

Qualitative coding, and especially grounded theory, has much in common with ML [8], especially in the way that theories (or models) are built from the data. We posit that QC can be used to create an immersive interface that facilitates human-assisted model building, using methods from QC to structure dialogue around the data that users must understand.

In such a system, users might give feedback that improves modeling accuracy by finding structure in the data, clarifying ambiguous items, and improving the quality of labeling overall. Giving domain expert users such an interactive ability should also increase their confidence in the system and its results.

3.1 How should QC structure interaction with ML?

Our goal is to increase the utility and accessibility of ML algorithms by making interaction with them understandable and efficient enough to allow domain experts without ML expertise to train them, and to explain their results to their peers. To achieve this, we will hide algorithmic detail with QC-based interaction focused around data, and with ML classifiers treated as collaborative coding partners.

Below, we sketch the specific challenges of human-ML interaction and how QC can help.

- *Recall*. Manually creating an ML training set is difficult, but evaluating much larger ML algorithm results is daunting, requiring users to recall and navigate connections between dozens of labels and thousands of examples (or more). We will implement QC-based codebooks (label indexes), data displays, memos and histories to make recall easier.

- *Error correction*. Most ML classifiers quite unrealistically assume that human-generated training sets are largely correct. QC does not make this assumption, relying on iterative reflection to correct errors. We will support reflection by displaying how well training examples ”cover” the data to code, how well training labels fit data features, and examples that significantly influenced the classifier.

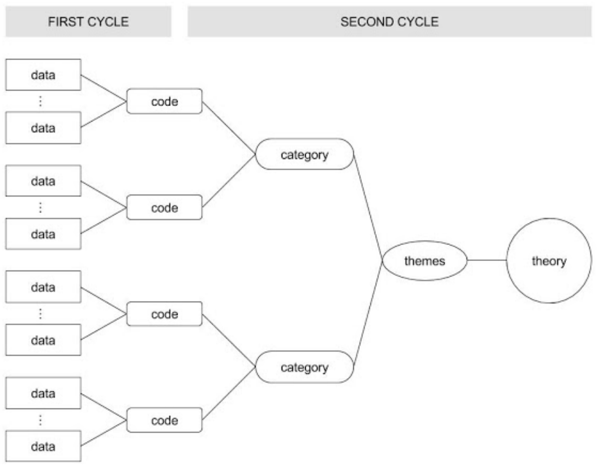


Figure 2: QC labels data with codes, groups codes into categories, and ultimately identifies themes and builds theories from the data.

- *Iteration*. ML training requires extensive iteration, and evaluating the results of each iteration is difficult, particularly for domain experts. QC supports manual iteration with improved measures of coding accuracy, and a focus on key data examples.

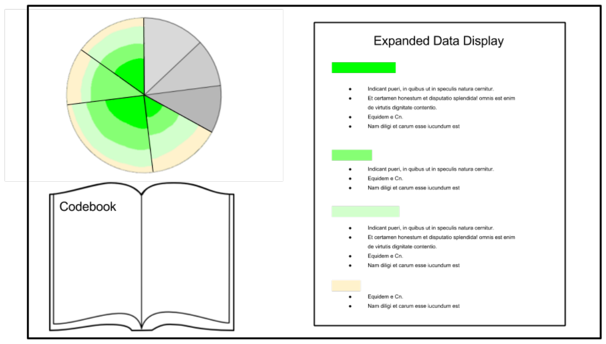
- *Collaboration*. For reasons of efficiency, reliability and trust, classification in many applied settings is intensely collaborative. Yet while ML has rarely considered human collaboration, QC has accommodated collaborative coding for decades. Using QC's displays and intercoder agreement measures, we will support the dialog of live partners, and treat ML-human interaction as dialog with an automated partner.

- *Efficiency*. ML often assumes that users already know how to label the training set, how to label it efficiently, and that they will label examples one data dimension at a time. QC includes grounded coding, with codes emerging as researchers encounter the data; and simultaneous coding, with researchers attaching multiple codes to each data item. We will bootstrap grounded coding with unsupervised learning, and support simultaneous coding/labelling during one manual data pass with parallel classifiers and hyperparameters. During that data pass, we will reduce the work required to achieve coverage by helping experts select key examples.

- *Interaction*. Many domain experts structure data by pushing paper representations of their data into piles - and many QC researchers still prefer this manual coding experience to the digital one offered by CAQDAS. Current ML systems cannot support such a natural interaction. We envision immersive tabletop and wall displays that reproduce this intuitive experience, allowing domain experts to create training sets and evaluate classification results.

4 An Immersive Interface for QC-structured ML

To reproduce the analogue data immersion that coders so often prefer, we believe the ideal ML UI will require three key elements:



- Indicant pueri, in quibus ut in speculis natura cernitur.
- Et certamen honestum et disputatio splendida! omnis est enim de virtutis dignitate contentio.
- Equidem e Cn.
- Nam diligi et carum esse iucundum est

- Indicant pueri, in quibus ut in speculis natura cernitur.
- Equidem e Cn.
- Nam diligi et carum esse iucundum est

- Indicant pueri, in quibus ut in speculis natura cernitur.
- Et certamen honestum et disputatio splendida! omnis est enim de virtutis dignitate contentio.
- Equidem e Cn.
- Nam diligi et carum esse iucundum est

- Equidem e Cn.
- Nam diligi et carum esse iucundum est

Figure 3: Possible Virtual Interface for a QC structured ML system.

- *Model visualization (Figure 3, top left).* An interactively updated visual summarizing how the model organizes the data overall. This summary helps users monitor their labeling/coding progress as they work with a data set that is much larger than in traditonal QC. Figure 3 uses a pie chart visualization, with colored slices representing (manually or automatically) categorized data, and gray slices uncategorized data provisionally grouped with unsupervised learning. As the user/coder works, colored slices grow and gray slices shrink. Each slice represents a category, with slice radial width representing the relative frequency of the category. The colors within each slice represent how well the model fits the data, with center green indicating good fit and peripheral yellow a poor fit (QC boundary cases). In the uncategorized gray slices, the UI highlights data items that it suggests the user classify next. These items would be most helpful for automatically classifying other items.

- *Derived data (Figure 3, bottom left).* Derived data include the codebook and history. The codebook display lists the most frequently used labels. Users can scroll to reveal other labels, or zoom in on a label to reveal descriptions, typical and boundary data cases, and corresponding category distributions. The history display lists the last few user actions. Users can scroll to reveal additional actions, with those that had significant impact on the model highlighted.

- *Data display (Figure 3, right).* A zoomed-in view corresponding most directly to QC's data displays. When a user selects a slice in the model visualization, the display shows significant data items in that slice, ordered and colored by model fit. Significant items may include those that are typical or boundary cases, and those that were particularly influential on the model. Users can select individual items in the model visualition or indeed the data display itself to reveal further detail (including analytic memos), and for labeling.

Training interaction might play out as follows: the user drags an unlabeled data point into the detail view to open the document. While looking at the document detail, she refers to the codebook to find the correct code or codes for the point, applying them to the point as she finds them. For the next data point, the user cannot find a matching code, and adds one to the codebook. The model visualization, derived data and data display change with each such action. This includes adding a slice to the pie chart, as well as changing the size and goodness of fit colors of the slices. The user then sees that the last several changes have not improved coverage and fit overall, and so unrolls those changes using the history.

Partnered QC depends on fostering a dialogue between coders, and in our scenario, we consider the underlying ML system to be a partner in the coding effort. Users have the freedom to explore the data at will, but the system - as the ML expert - needs to direct the labeling to foster the creation of a robust model. We have considered several possibilities for creating a helpful dialogue between machine and user:

- Suggest points to be labeled: the ML system should be able to highlight points that it believes will be most helpful for allowing auto labeling of other data points

- Ask for review of contentious labels: where system and user disagree on labeling, the system may ask the user to review and error correct if necessary

- Ask users to select human readable labels: if the system determines that a new code need be created, the user should be asked to provide wording that is descripitive to them

4.1 An Immersive Implementation

A purely virtual version of this ML-based interface could approximate the analogue immersive environments used by QC experts. Sitting around a large tabletop touch display, users might slide digital post-its or cards from the detailed view into the codebook or model visualization to label corresponding data items. They might tap those items to reveal detail or add analytic notes. As users work, the model displays its own changing labels to users in the same way: with virtual post-its that move like someone is pushing them. The interface shows the actions of remote coding partners in a similar way.

The next step might involve creating a two-way tangible interface, offering domain experts very familiar affordances. In this interface, the post-its and cards might be represented by physical post-its and cards. Users would be able to make a good guess at how to slide cards or post-its to categorize, and could pick up a card to reveal detail. The model would slide these same physical cards itself to represent its own work.

We might support such tangible input with specially printed cards that can be detected by a camera (e.g. with infrared ink fiducials). The same camera might scan and interpret handwritten notes written on these cards. Tangible output representing changes in the model are more challenging, but we might achieve it with simple Arduino robots, or with magnetic manipulation, like that used by high-end automated chessboards (e.g., [6]). Pangaro et al. [10] describe a similar approach.

5 A SUCCESSFUL INTERACTIVE ML — WHEN MIGHT WE SUCCEED?

To begin to understand how to structure a system that facilitates this dialogue between user and machine, we should consider the indicators of a successful system, and how to measure each.

One primary goal of the system we have described is to (ideally) eliminate the need for an ML expert to assist during the building and training of a model. Or, *can users without ML expertise steer ML behavior and produce robust results?* The obvious marker for a successful case is that a model built without the aid of an ML expert should preform at least as well as a model built by the ML expert. Models can be evaluated by any of the common measures for that type, such as accuracy for classification models.

The other major consideration in building this type of system is whether or not users will take advantage of this new capability. An established method for evaluating new technology is the Technology Acceptance Model [12], which is administered as a self-report questionnaire that asks potential users if they believe the technology is useful and easy to use. If users do not agree, then they are unlikely to adopt the new technology. Further goals for the system include group collaboration and inclusion of and consideration for non-domain experts as collaborators.

As with any human-facing system, we should consider standard usability heuristics [9] during design. Heuristics such as these provide standard guidelines for interface design that improve usability and more importantly, produce a familiar interface for users that reduces the initial learning curve and can increase adoption rates.

Our proposed QC/ML system should fulfill several of these heuristics. System status should always be visible via the codebook and the data display. The system assists with error recovery by maintaining a history supporting undo, and automatically evaluating feedback from the human users and highlighting poorly fitting data for further consideration. Rather than relying completely on user recall, recognition in the codebook and display should aid it, for example by reducing the likelihood of creating duplicate labels. Finally, the QC-based language and structure of the system should be easily understood by users, as it minimizes references to ML terminology, technology and components. Users should not feel as though they need to be knowledgeable in ML to use the system.

6 Conclusion

Human-in-the-loop machine learning systems have many advantages over traditional, automated ML. By allowing non-expert users to participate effectively in training a model, human-in-the-loop ML leverages the more flexible and accurate human pattern matching in complex, weakly structured domains; ultimately building more robust models with the need for fewer pre-labeled instances in the data.

We advocate qualitative coding as a promising way of structuring human-in-the-loop ML. It is a natural approach, having structured human classification for decades.

References

[1] S. M. Drucker, D. Fisher, and S. Basu. Human-computer interaction interact 2011. 6948, 1 2011.

[2] A. Endert, R. Chang, C. North, and M. Zhou. Semantic interaction: Coupling cognition and computation through usable interactive analytics. *IEEE Computer Graphics and Applications*, 35(4):94–99, July 2015.

[3] A. Endert, P. Fiaux, and C. North. Semantic interaction for sensemaking: Inferring analytical reasoning for model steering. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2879–2888, Dec 2012.

[4] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 473–482, New York, NY, USA, 2012. ACM.

[5] A. Endert, M. S. Hossain, N. Ramakrishnan, C. North, P. Fiaux, and C. Andrews. The human is the loop: new directions for visual analytics. *Journal of Intelligent Information Systems*, 43(3):411–435, 1 2014.

[6] Excalibur. Excalibur electronic 740d phantom force electronic chess set. `https://www.amazon.com/dp/B0018PWUJA/ref=cm_sw_r_cp_ep_dp_XfDCzbKXPM7JQ`, 2017. [Online; accessed 21-July-2017].

[7] J. A. Fails and D. R. O. Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.

[8] M. Muller, S. Guha, E. P. Baumer, D. Mimno, and N. S. Shami. Machine learning and grounded theory method: Convergence, divergence, and combination. In *Proceedings of the 19th International Conference on Supporting Group Work*, GROUP '16, pages 3–8, New York, NY, USA, 2016. ACM.

[9] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM, 1990.

[10] G. Pangaro, D. Maynes-Aminzade, and H. Ishii. The actuated workbench: computer-controlled actuation in tabletop tangible interfaces. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 181–190. ACM, 2002.

[11] J. Saldaña. *The coding manual for qualitative researchers*. Sage, 2015.

[12] V. Venkatesh and F. D. Davis. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science*, 46(2):186–204, 2000.

[13] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.