Building Bridges: A Case Study in Structuring Human-ML Training Interactions via UX

Johanne Christensen, Benjamin Watson
Department of Computer Science
North Carolina State University
{jtchrist,bwatson}@ncsu.edu

AJ Rindos, Stacy Joines
IBM
{rindos,joines}@us.ibm.com

Abstract

With the increasing ubiquity of artificial intelligence and machine learning applications, systems are emerging that require non-ML experts to interact with machine learning at the training step, not just the final system. These users may not have the skills, time, or inclination to familiarize themselves with the way machine learning works, so training systems must be developed that can communicate the necessary information and facilitate effortless collaboration with the user. We consider how to utilize techniques from qualitative coding, a human-centered approach for manual classification, and build better user experience for ML training.

Introduction

Technology has always been designed to assist humans in completing their tasks. As it advanced, particularly in computing, the types of tasks that technology could address shifted from simple (performing calculations or formatting text) to more complex (language understanding and image recognition). While complex AI applications are on the rise, their deployment still requires significant human effort, often by domain experts who have little understanding of how AI works. Thus when training a machine learning (ML) model, domain experts must work with ML experts, making ML solutions for many tasks too costly or simply infeasible. If we are to realize the full potential of machine learning (and, by extension, AI), we must begin building systems that can be deployed or at least improved by end-users.

User experience (UX) seeks to create products that are not only useful and usable, but also motivating and pleasurable. Toward this end, UX practitioners employ a human-centered practice that we believe is essential for building AI or ML solutions useful by domain experts without ML familiarity. Such solutions must communicate effectively with users about their domain, without bogging them down in ML detail.

Classifier Training for non-ML Experts

Even when ML expertise is unnecessary or can be minimized, the requirement of extensive domain expertise can be a significant barrier to adoption. In specialized domains such as law or finance, expert time is expensive. Methodologies that reduce the required number of labeled examples may not be enough: very large datasets can still make the labeling task impractical.

Nevertheless, complete automation of the training process is not only infeasible, but also inadvisable. Human oversight creates trust in the model once it is deployed.

Qualitative Coding

Qualitative coding (QC) (Saldaña 2015) is a manual classification method commonly employed in the humanities and behavioral sciences to extract meaning from data such as text, imagery, and video. The subset of QC called grounded theory particularly has much in common with traditional machine learning (Muller et al. 2016), especially in the way that theories (or models) are built from the data. With this in mind, we posit that QC can form an interface for ML training, facilitating interaction and structuring dialogue around the data for analysis. QC may be ideal for building an ML platform that domain experts can comfortably use, without significant ML expertise.

(Shneiderman 1982) argued that direct manipulation in a desktop interface provides a better experience for non-expert computer users. Similarly, we believe that QC supports direct manipulation of data, creating an intuitive yet still efficient experience. QC practitioners commonly use note cards or post-its to represent data, physically grouping post-its to cluster data points. Attaching meaning to movement in the ML training interface should increase communication bandwidth, allowing domain experts to communicate about their data not only linguistically, but also behaviorally.

Insights from the Field

We have recently conducted interviews with developers building custom interfaces for clients to train classification models on unstructured text documents. In these systems, out of the box, the ML software is able to classify documents with limited accuracy. For improved accuracy, domain experts must train the system further by labeling data.

Our interviews have surfaced several pain points during ML training. Interaction quickly becomes repetitious, with experts providing feedback about the classification of several data items, iteration after iteration. Developers observed that users find labeling documents burdensome.

One method of mitigating this tedium is to simplify the feedback task: instead of asking how the document should be classified, users might indicate only whether the current label is correct. Yet this requires a system that can:

- identify documents that likely should have the current label
- decide which of those documents need manual labeling, ie. which examples, once manually labeled, will most improve the model. Indeed, such a capability might also be used to reduce the number of feedback task iterations domain experts must perform.

When relevance is more nuanced, answering a yes/no question about correctness might be too simple to produce a good model. Ranking of relevance might be a good compromise, but as feedback complexity grows, so does subjectivity between coders. How should a model account for the possibility that two domain experts may have differing ideas on relevance? In QC methodology, memos allow coders to compare notes on why they picked certain codes, and statistical intercoder agreement scores measure overall cross-coder consistency.

Another way of reducing the tedium of feedback is to vary the task, and spread it across multiple expressive modalities (e.g. language, behavior, vision and sound). We believe that the movement and highly visual nature of QC coding and its data displays will be quite helpful in this regard.

Nearly all developers mentioned the difficulty of communicating the confidence the ML model has in its classifications to non-ML experts. Domain experts should prioritize feedback about high confidence classifications that are wrong, and about lower confidence classifications in general. Domain experts often misinterpreted confidence scores, believing that they were being provided with classifications already known to be incorrect. This often led them to provide inaccurate feedback.

The obvious solution is to train users about confidence scores. However, a domain expert who specializes in a field that does not regularly utilize probabilities neither needs nor wants to learn what confidence scores mean. Instead, as one developer observed, rather than asking domain experts to focus on certain documents based on classification confidence, we might highlight those documents in display (and perhaps filter out documents not needing attention, again reducing feedback task iterations).

How can QC structure interaction with ML?

Our goal is to increase the utility and accessibility of ML algorithms by making interaction with them understandable, efficient and engaging enough to allow domain experts 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 explore how QC addresses (Nielsen and Molich 1990)'s usability heuristics.

- *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). QC-based codebooks (label indexes), data displays (using note cards and post-its), memos and histories help domain experts remember and navigate through such large collections of information.
- *Error correction*. Errors during model building can come from either the human or the ML. QC relies on iterative reflection to correct errors. Data displays provide the context in which errors can be identified, while using displays to communicate how well training examples cover the data, how well training labels (or classes) fit data features, and examples that significantly influenced the classifier provide multiple opportunities for finding and resolving errors.
- *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. Labeling in iterations also breaks the task into more manageable pieces, mitigating fatigue, attention drift and stress for the user.
- *Collaboration*. For reasons of efficiency, reliability and trust, classification in many applied settings is intensely collaborative. Collaboration in QC is inbuilt, supporting the dialog of live partners with displays and intercoder agreement measures. The ML is considered an additional partner, so providing human-ML interaction on par with human-human interaction is essential.
- *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. These techniques also support users learning the task as they perform it, as it is flexible enough that novices to the training task itself can effectively interact with the system from the start. Even during practice, users can contribute to training, even if some of their input needs to be changed later.
- *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 qualitative coding software. Current ML systems cannot support such a natural interaction. We envision tabletop and wall displays that reproduce this intuitive experience to allow domain experts to create training sets and evaluate classification results.

Structuring a Dialogue Beyond Words

In a human-machine partnership, what should the interaction look like? QC methodologies are often described as structuring a dialogue around and about the data. Extending that metaphor into a design consideration provides a strong foundation for building ML software that helps non-expert users

work in partnership with the machine to complete their desired task.

Although it is common to consider a dialogue as an exchange of words, communication around a subject need not be linear, synchronous, or restricted to language. For example, using visual representations to summarize data or nonverbal behaviors to convey ancillary information are alternative techniques that can carry a dialogue without human (or computer) language.

With this in mind, we consider what a dialogue about training a classifier might look like. Crucially, it is vital that the system is structured so that the machine communicates at a human level, accommodating users at all levels of technical expertise. In this setup, the burden of driving the conversation and managing the task remains on the system, but the user remains the ultimate authority, with the final say should there be a dispute. Table 1 shows a possible task breakdown between the human and machine partners.

We imagine an interface where the system continually communicates the state of the model in training. This could be via visualizations of the progress of the model as it approaches a trained state. In addition to providing information to the user, a complete solution for communication requires a system that can properly decode the user's state to fully understand the user's actions. For example, if the user hesitates when labeling a data point, the system might learn that this behavior means the label should be applied with a lower confidence score.

Human Partner	ML Partner
labels docs	gives feedback on model
manages the knowledge	manages the data
corrects errors	finds possible errors
	manages task breakdown
	learns from user's actions

Table 1: Task breakdown between Human and System

Consider an ML classifier that groups data items into two categories: relevant and irrelevant. In a tabletop display that reproduces QC interaction, a domain expert is training the ML. A visualization shows that the ML model has not yet classified a third of the data, and that another 5% of the data is poorly fitted. Directly in front of the expert, dozens of data items are represented by cards. Two piles of cards are labeled "relevant" and "irrelevant," another "unlabeled," and a fourth "revisit."

The expert drags one of the unlabeled cards to the center, where it expands, showing additional detail. She considers for a moment, then swipes the card rapidly toward the "relevant" pile. The card spins and curves on its way to the pile where it settles in with an audible plop, and the visualization updates as a result, with only a quarter of the data still unclassified. The expert then brings another unlabeled card to the center. This data item is more difficult, and she consults her own and others' memos in the QC codebook before swiping the card to the "irrelevant" pile. The classifier notes the expert's hesitation and marks the labeling of this data item as "uncertain."

Bridging the Disconnect between Humans and Machines

Understanding how humans communicate with machines is key to building effective interactive systems (Suchman 1987), like those being developed for ML classifier training. UX's human-centered approach is particularly valuable in this regard. For example, the human tendency to anthropomorphism can be leveraged to enrich human-machine interactions (Levillain and Zibetti 2017)(van Allen 2017). By treating human-machine interaction as a form of human-to-human communication, we might improve interaction, particularly for non-technical users. Today's technology may finally be enabling systems that realize this vision and the vision of affective computing (Picard and Picard 1997): perceiving, understanding and expressing – communicating – with users not just by language and example, but by behavior and emotion. This sort of interaction is rapidly becoming necessary as human machine dialog enters all phases of our daily lives and indeed our lifetimes.

Acknowledgements

This work was partially funded by an IBM Faculty Award.

References

Levillain, F., and Zibetti, E. 2017. Behavioral objects: the rise of the evocative machines. *Journal of Human-Robot Interaction* 6(1):4–24.

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 19th International Conference on Supporting Group Work*, GROUP '16, 3–8. New York, NY, USA: ACM.

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

Picard, R. W., and Picard, R. 1997. *Affective computing*, volume 252. MIT press Cambridge.

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

Shneiderman, B. 1982. The future of interactive systems and the emergence of direct manipulation. *Behaviour & Information Technology* 1(3):237–256.

Suchman, L. A. 1987. *Plans and situated actions: The problem of human-machine communication*. Cambridge university press.

van Allen, P. 2017. Reimagining the goals and methods of ux for ml/ai. *AAAI Spring Symposium Series*.