Open Coding for Machine Learning Data

Magdalena Price [1] **Dylan Hadfield-Menell** [1]

Abstract

Data-driven decisions have an undeniable influence on our quality of life, and promises of fair-decision making often fall short, resulting in unintended harms. Large datasets in particular fall victim to this phenomenon, as the time and cost associated with analysis demand less thorough data evaluation. Through Open Coding for Machine Learning, we approach this problem from the perspective of creating a better-quality data set, working to reduce unfairness by addressing goal misspecification. Our methodology combines data analysis techniques outlined by Kathy Charmaz in *Constructing Grounded Theory* with machine learning techniques, presenting a novel annotation interface that allows a single annotator to effectively devise custom labels for a large, unlabeled dataset. This approach will make open-ended data analysis more scalable, enabling qualified individuals to annotate important collections of data to be used in machine learning prediction models.

1. Introduction

Data-driven decisions made by models are undeniably influential in modern life. They have the potential to do much good, but also the potential to do much harm (Aronova et al., 2017). Predictive models such as COMPAS (Inc., 2012) promise to more fairly identify high-risk, violent re-offenders, but their reliance on historical data means the predictions they make perpetuate the same discriminatory behaviors they were made to counteract (Angwin et al., 2016). This is a common theme in critiques of predictive models, particularly when considering the fairness of machine learning predictions (Obermeyer et al., 2019) (Faliagka et al., 2014).

*Equal contribution [1]Algorithmic Alignment Group, Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Magdalena Price <maprice@csail.mit.edu>, Dylan Hadfield-Menell <dhm@csail.mit.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Work in fairness has taken several different approaches to solving this problem, from fine-tuning the input features of prediction algorithms to pre-processing datasets to reduce potential contributors to bias (Zhou et al., 2017). These approaches tackle the problem of biased results given a fixed, sub-optimal dataset. An alternative approach is to consider the ways in which one can create quality datasets. In this work, we research the latter, building on studies that consider unfairness from the perspective of **goal misspecification** (Mullainathan & Obermeyer, 2021).

With goal misspecification, predictive models for subjective observations are trained on correlative metrics ("labels") that may not accurately reflect the nuanced nature of what is being predicted. This problem is particularly significant in the case of large datasets, as obtaining accurate labels for a large dataset requires both creativity and domain expertise (Boecking et al., 2020), which is expensive and time-consuming. As a result, alternative, less thorough methods of labeling are utilized. Some examples include crowdsourcing, which is expensive and runs the risk of validating un-qualified interpretations (Zhou et al., 2017) (Gao et al., 2011), and data programming, which is scalable but difficult to apply to real-world problems (Ratner et al., 2016). Thus, it is still hard to curate custom labels for large, real-world datasets.

An additional problem is how to effectively structure the initial interactions with the data. In some cases, there is a clear idea of what labels you want to create, but that is not always the case. Regardless, it is the user's job to make hypotheses about patterns in the data and to generate effective labels rather than relying on pre-determined notions of appropriate labels. For this problem, we look to grounded coding theory (Charmaz, 2014) as a starting point.

The approach we take is to research currently effective data-labeling processes and expand upon them, improving their scalability, and using them to address the aforementioned issues. We combine modern-day human-led data labeling practices with semi-supervised machine learning techniques to provide an interface that is both human-focused and human-scalable.

We present our work - Open Coding for Machine Learning[1]

[1]Project code can be found at https://

- with the goal of facilitating systematic data auditing and intentional label curation.

We design an interface that integrates a language model into Kathy Charmaz's coding methodologies (Charmaz, 2014), allowing a user to better analyze and better understand the social realities of the data at hand. To start, our interface guides a user through an initial, open-ended investigation of the data with free-form annotation. Then, we allow them to search through these annotations and develop an initial dataset of examples. The final step of the process is semi-supervised learning, where we train a model on a stream of labels, tracking accuracy and allowing the user to indicate completion when they are satisfied with the model's accuracy.

2. Related Work

Currently-existing services that digitize open coding, such as **ATLAS.ti** (https://atlasti.com/) (Atlas.ti, 2022) and **Saturate** (http://www.saturateapp.com/) (Sillito & De Alwis, 2009), offer many of the same tools that we aim to offer, and as more fully-fleshed out products, offer more features. However, we wanted to create an *open-source platform* for digital open-coding without cost barriers. We also wanted to create a more scalable platform, motivating us to create our own interface which we could integrate with machine learning techniques, rather than building from currently-existing open-source coding projects (Sillito & De Alwis, 2009). Regardless, these services provided helpful insight into the types of UI features we might want to support.

3. Open Coding for Machine Learning

Open Coding for Machine Learning is an interface designed to help make effective data labeling scalable by 1) borrowing from social science data analysis techniques and 2) leveraging semi-supervised machine learning to expedite the process of data labeling. Throughout this paper, we will showcase an implementation that uses the text samples from HappyDB (Asai et al., 2018), but the interface can currently support any text-based dataset.

3.1. Grounded Coding Theory

Kathy Charmaz's *Constructing Grounded Theory* (Charmaz, 2014) is an extended review and analysis of qualitative research strategies. In particular, it outlines **grounded theory coding**, describing it as a process that can help an individual deepen their understanding of not only the dataset being evaluated, but also of the underlying issue being defined. The higher level process can be divided into two successive parts - **initial coding** and **focused coding**.

Initial coding facilitates an exploratory process, allowing an individual to observe nuances and gain a deeper understanding of the data. We derive inspiration from one subset of initial coding, **line-by-line coding**, as it is scalable, thorough, and most similarly models the desired format of many machine learning prediction projects. Figure **??** presents an example of line-by-line coding, where free form annotations are written for each line of text. In addition to encouraging the thorough exploration of the dataset at face level, this process encourages introspective thinking about one's own interpretation of the dataset.

The next part of the process, focused coding, builds on an individual's gained knowledge of the dataset and previously-defined codes, facilitating the analysis, improvement, and re-grouping of previously-defined codes. Charmaz specifically describes the higher-level process of focused coding as "concentrating on what your initial codes say and the comparisons you make with and between them" (Charmaz, 2014). Figure **??** visualizes our interpretation of this statement, which was used to motivate our design. We understood focused coding to be the process of analyzing the corresponding annotations for each line, and forming a cohesive and comprehensive set of more focused categories to describe the annotations written. This process allows for not only pattern-finding, but also careful consideration of what is truly *valuable* in the specific context of the information at hand.

3.2. Transforming Open Coding into an Interface

Figure 3 shows our approach to integrating grounded coding theory (see 1) and machine learning (see 2). The lower part of the figure also indicates the relevant page names of the interface for each interaction implemented.

The first phase of the design process was translating Kathy Charmaz's description of line-by-line coding (Charmaz, 2014) into a page that was intuitive and efficient. The Open Coding page (Figure 4) provides a space for users to annotate lines of text freely and efficiently, thanks in part to the implemented hot-keys **tab** and **shift-tab**, which allow a user to type entries without leaving the keyboard.

Following, the Assisted Grouping page (Figure 7) implements the foundational ideas behind focused coding (Charmaz, 2014), allowing the user to freely view and group their annotations into more focused categories.

Lastly, the Verification page (Figures 5 and 6) serves as an invaluable connection between the open coding process and standard machine learning model training. Specifically, this page implements a feedback loop between the predictive model and the user, intrinsically valuing the human's input

github.com/Algorithmic-Alignment-Lab/OpenCodingForMachineLearning.

more than the model's prediction.

3.3. Scaling Open Coding

A significant portion of this research leverages pre-existing implementations from the **Hugging Face Transformers** (Wolf et al., 2020) library. In particular, this research builds on and inherits from **DistilBertForMaskedLM** and **DistilBertForSequenceClassification**, utilizing the former's forward method for pre-training and the latter's forward method for fine-tuning. While we chose to use **DistilBert** implementations for version 1, the ideas behind Open Coding for Machine Learning can be implemented using a variety of language model types, and the higher-level ideas are not restricted to DistilBert alone.

To pre-train our model, we borrow the forward method, related helper functions, and related helper classes from DistilBertForMaskedLM to run Masked Language Modeling (MLM) on the unlabeled text corpus. This form of unsupervised training allows our model to gain familiarity with the dataset by adapting its feature embeddings to the desired text corpus. This was implemented in order to improve the scalability of our approach by improving the model's initial predictive effectiveness. From the perspective of our interface, this step occurs at the beginning, before the Open Coding page interactions (Figure 4).

To fine-tune our model, we borrow the layers in the forward method of DistilBertForSequenceClassification, integrating the predictions into the feedback loop previously mentioned in the Verification page (Figure 6). From the reference frame of the interface, the first fine-tuning session is run after the user successfully submits their initial groupings from the Assisted Grouping page (Figure 7). Following, the model is trained using the grouped text items, and returns a randomly-selected group of predictions to the Verification page. Once the user completes a verification round, they can either re-train the model with the additional labeled examples, or they can continue, asking the model to label the rest of the dataset.

3.4. Results

We asked five lab members and academic colleagues over the course of two weeks to do a shortened demo of our interface, providing us with not only usability feedback, but also feedback on the higher level ideas behind Open Coding for Machine Learning.

Our demoers found the interface fairly intuitive, but provided several places for UI improvements, enabling our interface to go through several iterations of design between demos. Continuing, the process of initial coding (Figure 4) followed by focused coding (Figures 7 and 5) was well-received by our demoers, and our participants were able to create reasonable labels and gain a deeper notion of happiness by analyzing HappyDB (Asai et al., 2018) through our interface.

While the digitizing of the coding process was fairly successful, the model was not able to reach a high-enough accuracy efficiently enough to confidently scale the coding process. As shown by Figure 8, the accuracy of the predictive model was strongly associated with the amount of training data provided, and pre-training did not offer any accuracy "head-starts". Thus, the process is currently not scalable, as several rounds of verification would be required to achieve accurate results.

If the model was able to reach $> 95\%$ accuracy with just 5% of the full HappyDB dataset being labeled, then it would take approximately 100 rounds of Verification with 50 examples per round to achieve the desired accuracy (about 800 minutes of work, estimated). We found during our demos that the specific efficiency of the interface was highly dependent on the nature of the user, but on average, a verification round took 6-10 minutes to start, and took less time as the round numbers increased.

Currently, the interface requires more than 5% of the dataset to be labeled to achieve reasonable accuracy. Regardless, we believe that current machine learning methodologies are near optimal enough to support effectively scaling this interface, and we explore alternate machine learning techniques in Future Work (4).

4. Future Work and Limitations

4.1. Limitations

One of the drawbacks of our evaluation was that, in order to be respectful of our lab members' time, we had to do a shortened demo of our interface using our local machine and approximately 14,000 examples pulled from the HappyDB dataset. As a result, we can only estimate the amount of time and computational power required for analyzing the full HappyDB dataset based on these demos and our personal setup.

In the future, we would like to a more formal human study on the entire database, designing it as either an individual study or a group study, exploring the ways in which we can encourage productive collaboration through our interface. We also want to enable hosting on participant's machines, in order to test the compatibility of our interface with varying computational setups.

4.2. Future Work

A central goal of our research was to create a foundational framework that was easily expandable, such as to support future research in this area.

We have currently implemented the OpenCodingModel with DistilBert (Wolf et al., 2020) as its base, as it is efficient while still being effective as a language model. However, the Open Coding framework can be repeated with other model types, and is not just limited to DistilBert or Bert-type models. The interface model's accuracy could be improved by the selection and integration of a different model type, and this is an exercise we leave to future research.

We might also think about improving our model's accuracy by applying active learning techniques, such as showing the user predictions that the model is less confident about overall, and thus encouraging the effective utilization of the time spent in the verification feedback cycle. Another interesting active learning approach could utilize the pre-training loss to identify which subsets of the text we should prioritize showing to the user (Yuan et al., 2020), both in the Open Coding stage (Figure 5) and the Verification stage (Figure 4). This would perhaps help us more effectively utilize not only the time spent in the feedback cycle, but also the time spent annotating.

We might also imagine re-vamping the Verification (Figure 5) user flow entirely, instead asking the user to input a label and giving them text examples that the model has predicted to have that label. This could be an effective way to allow the user to prioritize training for specific labels that they feel are not being understood well enough by the model.

5. Concluding Statements

Thus, we see great potential in approaches that seek to improve machine learning practices by re-prioritizing humans in the process, and we believe that such approaches can achieve both effectiveness and scalability. We conclude with the continued belief that deriving inspiration from human-led data labeling practices and augmenting them with machine learning techniques is a promising approach for supporting the effective, scalable labeling of large datasets.

Acknowledgements

We would like to extend our deepest gratitude to members of the Algorithmic Alignment Group for supporting and evaluating our research. We would also like to thank our fellow academic colleagues for demoing our interface, providing invaluable feedback.

Additionally, we would like to thank members of the MIT Anthropology Group for their help during the design process, as they identified aspects of the coding process that we had overlooked, and suggested functionality that would improve upon Open Coding for Machine Learning.

Lastly, we thank our family and loved ones for their continued support, guidance, and advice.

References

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. How We Analyzed the COMPAS Recidivism Algorithm. *ProPublica*, 2016.

Aronova, E., Oertzen, C. v., and Sepkoski, D. Introduction: Historicizing Big Data. *Osiris*, 32(1):1–17, 2017.

Asai, A., Evensen, S., Golshan, B., Halevy, A., Li, V., Lopatenko, A., Stepanov, D., Suhara, Y., Tan, W.-C., and Xu, Y. HappyDB: A Corpus of 100,000 Crowdsourced Happy Moments. In *Proceedings of LREC 2018*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

Atlas.ti. Qualitative Data Analysis, 2022.

Boecking, B., Neiswanger, W., Xing, E., and Dubrawski, A. Interactive Weak Supervision: Learning Useful Heuristics for Data Labeling. *arXiv preprint arXiv:2012.06046*, 2020.

Charmaz, K. *Constructing Grounded Theory*. SAGE Publications Sage UK: London, England, 2014.

Faliagka, E., Iliadis, L., Karydis, I., Rigou, M., Sioutas, S., Tsakalidis, A., and Tzimas, G. On-line consistent ranking on e-recruitment: seeking the truth behind a well-formed cv. *Artificial Intelligence Review*, 42(3):515–528, 2014.

Gao, H., Barbier, G., and Goolsby, R. Harnessing the Crowdsourcing Power of Social Media for Disaster Relief. *IEEE Intelligent Systems*, 26(3):10–14, 2011.

Inc., N. COMPAS Risk & Need Assessment System, 2012.

Mullainathan, S. and Obermeyer, Z. On the Inequity of Predicting A While Hoping for B. In *AEA Papers and Proceedings*, volume 111, pp. 37–42, 2021.

Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019. doi: 10.1126/science.aax2342. URL https://www.science.org/doi/abs/10.1126/science.aax2342.

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data Programming: Creating Large Training Sets, Quickly. *Advances in neural information processing systems*, 29:3567–3575, 2016.

Sillito, J. and De Alwis, B. Saturate: A collaborative memoing tool. In *Proceedings of UBC's First Annual Workshop on Qualitative Research in Software Engineering*. Citeseer, 2009.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Yuan, M., Lin, H.-T., and Boyd-Graber, J. Cold-start Active Learning through Self-supervised Language Modeling. *arXiv preprint arXiv:2010.09535*, 2020.

Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.01.026. URL https://www.sciencedirect.com/science/article/pii/S0925231217300577.

A. Figures

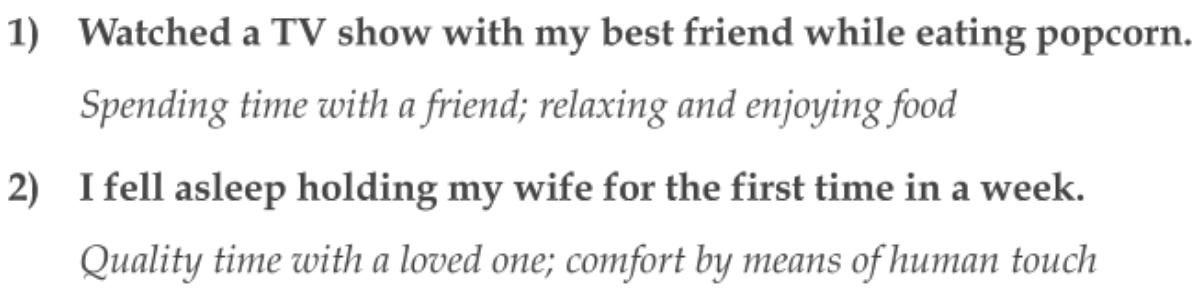

Figure 1. **Initial Coding - Line by Line**. The given figure visualizes an approach to line-by-line open coding, with each number being followed by a line of text, with a corresponding annotation beneath. In this data analysis technique, an individual reads each line independently and notes any desired thoughts. As an exploratory and contemplative process, it encourages through evaluation of the material at hand.

Figure 2. **Focused Coding**. The given figure visualizes an approach to focused coding, after taking the line-by-line coding approach. While there are many possible interpretations of focused coding, at it's core it is a methodology meant to help understand the breadth of the data, focusing in on the most important qualifying characteristic to form meaningful and comprehensive labels.

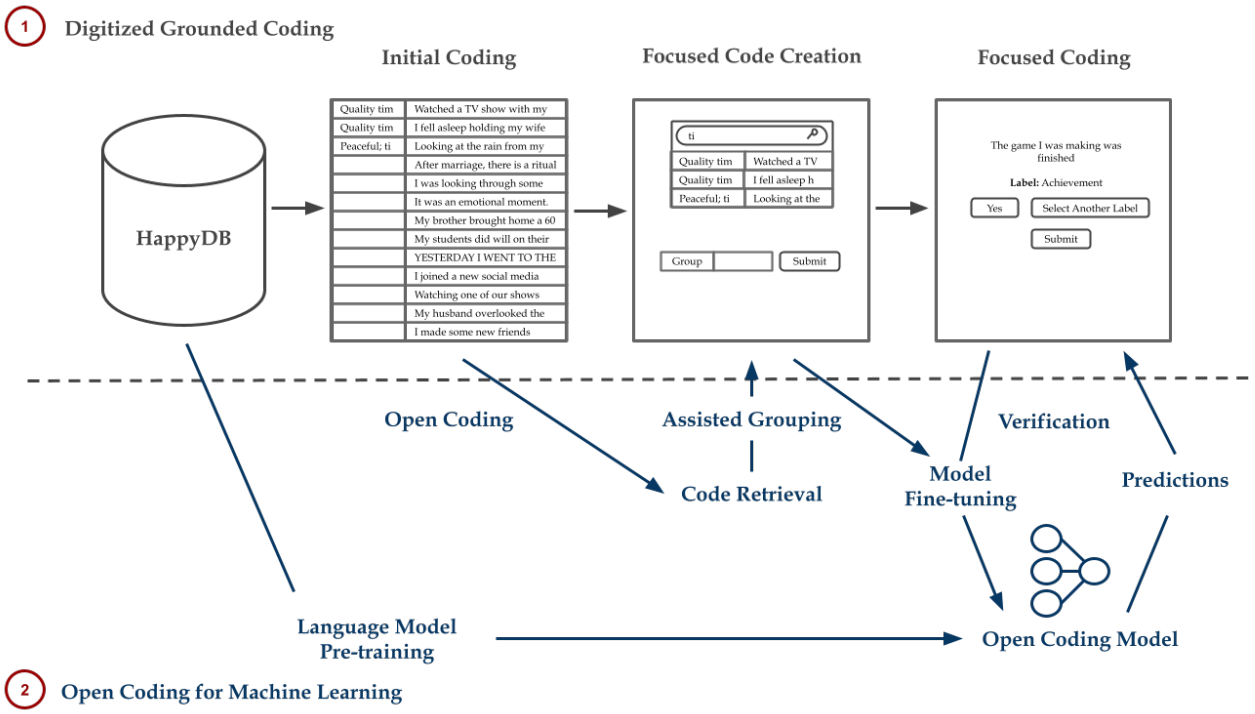

Figure 3. **Overview of Open Coding**. **Section 1** overviews a digitized version of initial coding and focused coding (Charmaz, 2014). During *initial coding*, we provide a space where an individual can free-write thoughts, line-by-line, about pieces of text from a corpus (HappyDB (Asai et al., 2018)). Those thoughts are used in the *focused code creation* process, where an individual is able to gather their thoughts and consolidate them into more focused categories. Lastly, *focused coding* presents predictions made on un-seen portions of the dataset, and enables an individual to validate or adjust the predicted category. **Section 2** indicates the corresponding Open Coding for Machine Learning interface pages and overviews techniques used to make the process more scalable, including the integration of a language model and code search functionality. Note that the model used for predictions is first pre-trained on the entire un-labeled text corpus, and is then fine-tuned as the individual continues to make and confirm category assignments (see 3.3). In order to better mimic the ability to re-reference previously defined codes in open-coding, we also include a code retrieval feature.

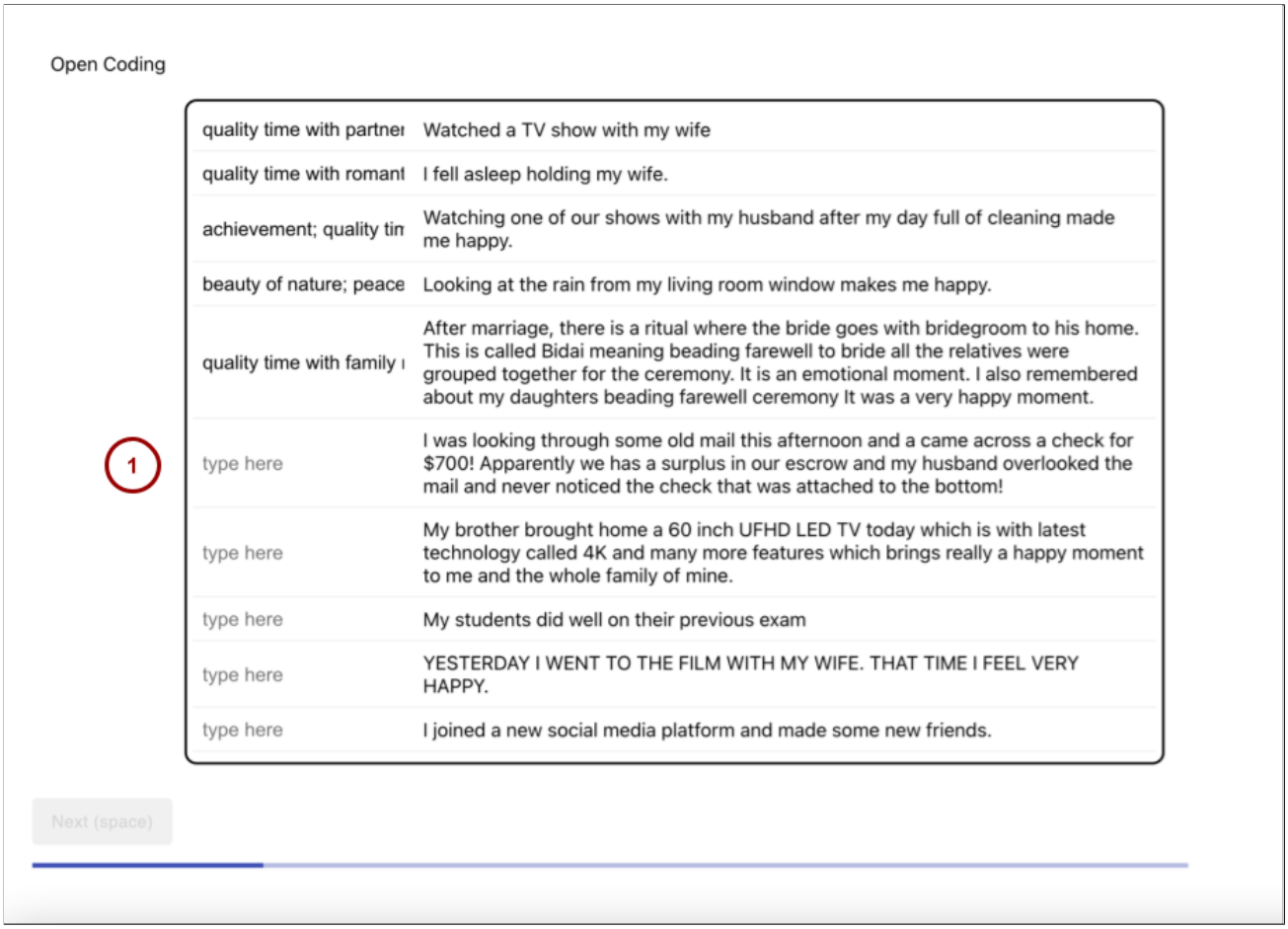

Figure 4. **Open Coding**. The user can freely annotate each given text example. The dual text-box/text view is scrollable, and can be navigated using the hot keys **tab** and **shift-tab** for ”down” and ”up”, respectively. All text examples must have a non-empty annotation before the ”Next” button is made available.

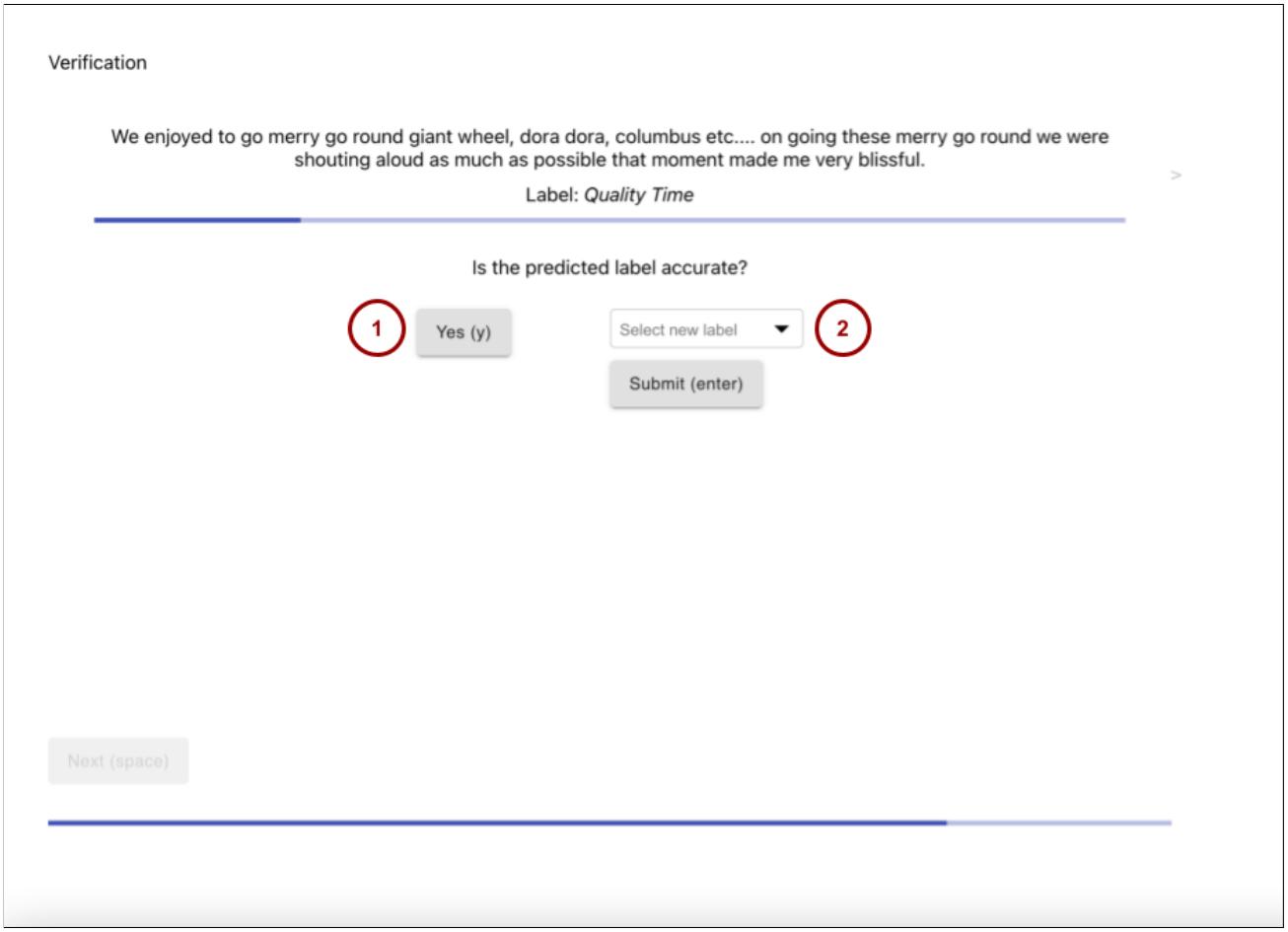

Figure 5. **Verification A**. The user can indicate that a predicted label is accurate (see 1), or they can assign a more accurate label by selecting an option from the dropdown and pressing "Submit" (see 2). Interactions are strictly linear, thus once a user has selected an accuracy for a particular prediction it cannot be undone. This implementation was chosen with the intent to provide a strictly guided experience, but can limit a user's ability to correct mistakes. Once all predictions have been evaluated, the user moves on to Stage B.

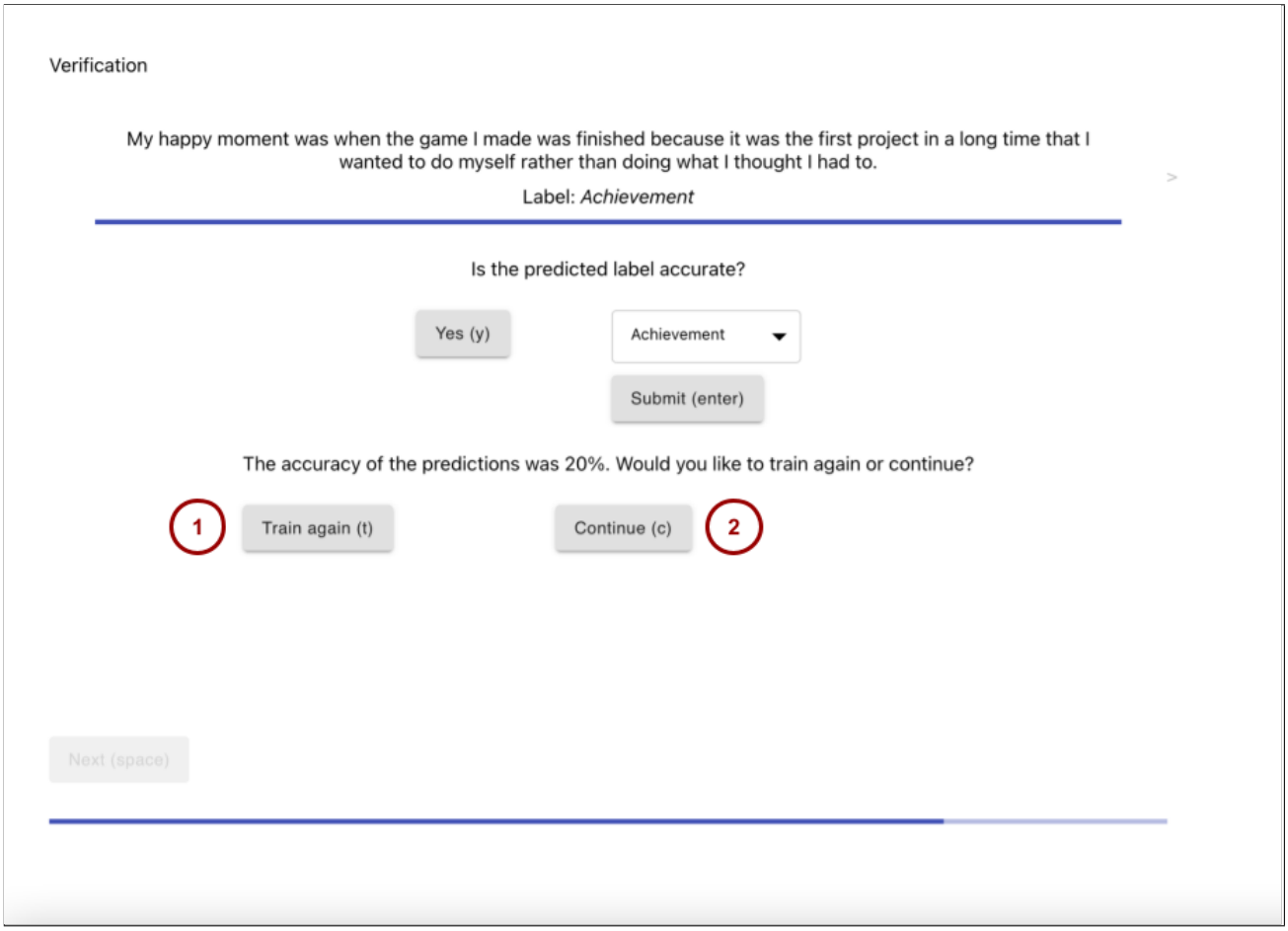

Figure 6. **Verification B**. The user can decide to train the model again on the previously-evaluated predictions, and receive more predictions to evaluate (see 1), or the user can continue (see 2), moving onto the results section and training the model but not requesting more predictions to evaluate. The "Next" button is made available if the continue button is pressed. Otherwise, the user returns to Stage A. Future iterations of this interface might consider hiding the continue button until a particular accuracy or certain average accuracy is achieved.

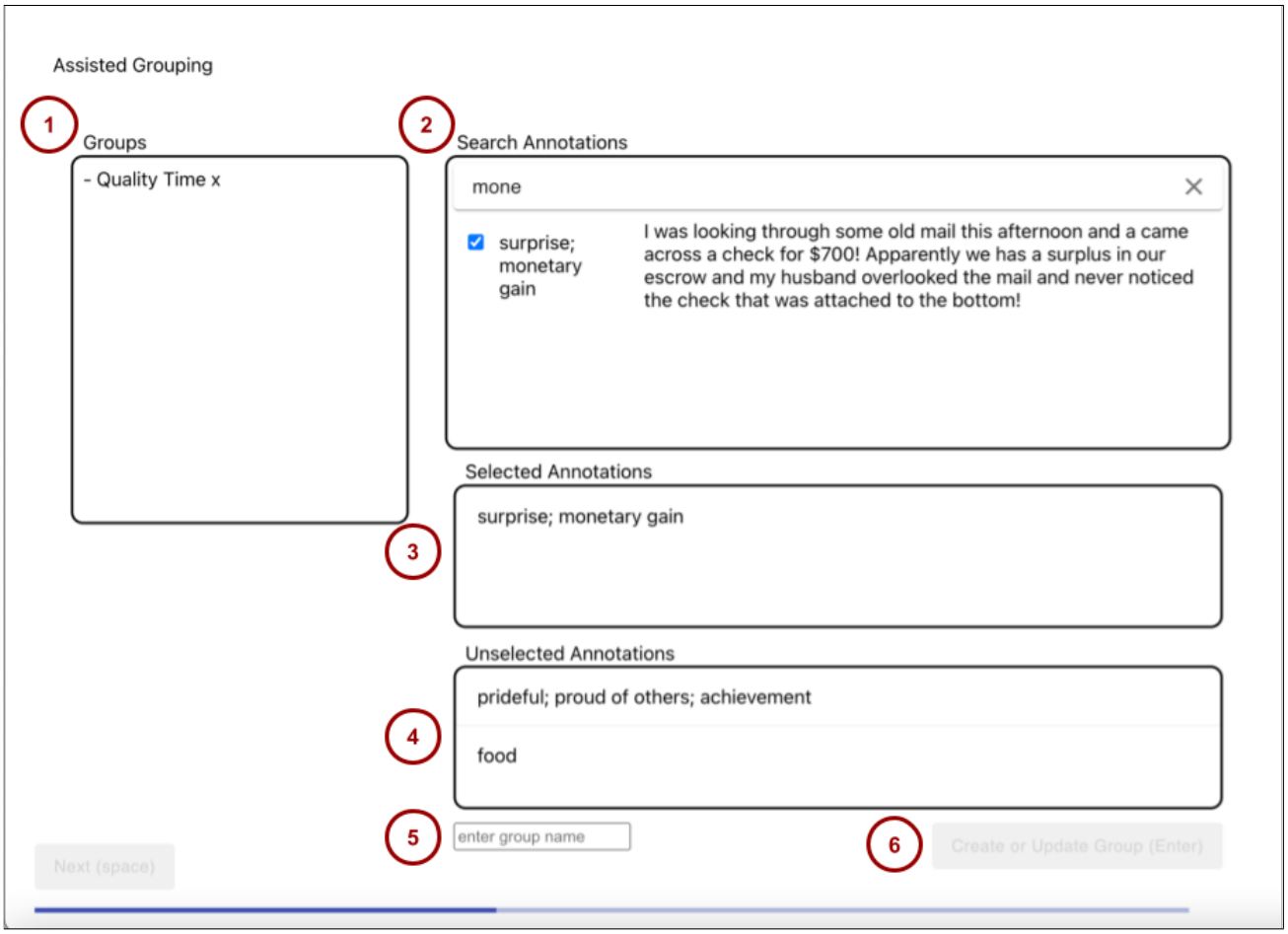

Figure 7. **Assisted Grouping**. The user can search the annotations they've written (see 2) and the lowercase-substring-matches will appear within the corresponding, scrollable view box. When elements are selected via their respective checkbox, they will appear under the "Selected Annotations" view box (see 3) and be removed from the "Unselected Annotations" view box (see 4). Once a user has selected at least one annotation, they may type into the group name text entry field (see 5), and they may create or update a group by clicking the relevant button (see 6). Groups will appear under the "Groups" view box (see 1), and they are expandable, deletable, and updatable. Once every annotation has been placed into a group, the "Next" button is made available. This page implements approximately all of the features we wanted to support, but does have some usability issues noted in Results (3.4).

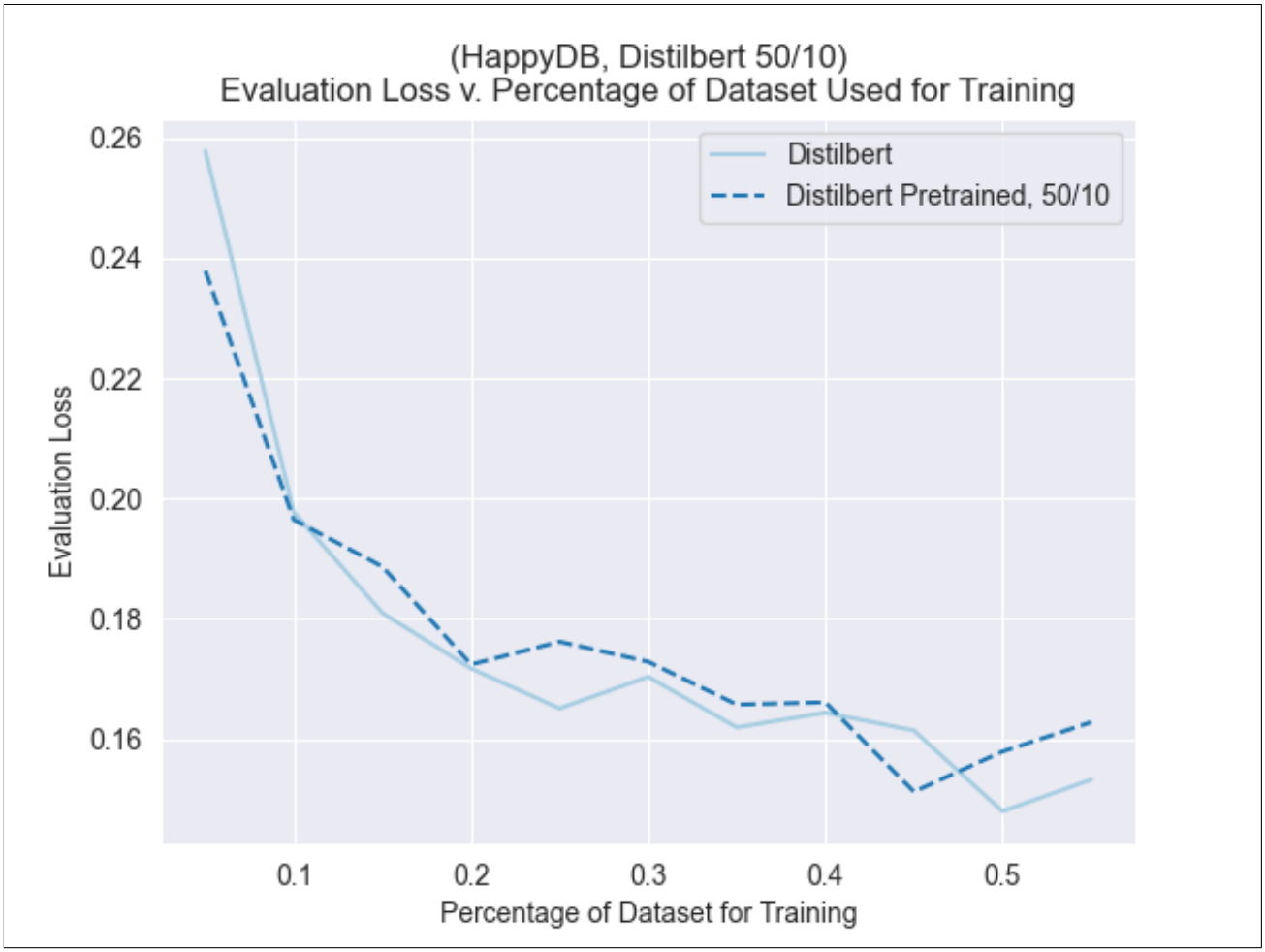

Figure 8. **Pretrained (50/10) Distilbert Performance - HappyDB**. Evaluation loss over percentage of the labeled data used for fine-tuning for OpenCodingModel (Distilbert) pretrained on HappyDB with batch size 50 and 10 epochs. As shown, the performance increase of pretraining is negligible, and perhaps even causes a decrease in overall performance.