Does Active Learning Reduce Human Coding?: A Systematic Comparison of Neural Network with nCoder

Jaeyoon Choi[1][0000-0002-8893-7898], Andrew R. Ruis[1][0000-0003-1382-4677], Zhiqiang Cai[1][0000-0002-2107-3378], Brendan Eagan[1][0000-0002-9189-1747], and David Williamson Shaffer[1][0000-0001-9613-5740]

[1] University of Wisconsin-Madison, Madison, WI 53706, USA
jaeyoon.choi@wisc.edu

Abstract. In quantitative ethnography (QE) studies which often involve large datasets that cannot be entirely hand-coded by human raters, researchers have used supervised machine learning approaches to develop automated classifiers. However, QE researchers are rightly concerned with the amount of human coding that may be required to develop classifiers that achieve the high levels of accuracy that QE studies typically require. In this study, we compare a neural network, a powerful traditional supervised learning approach, with nCoder, an active learning technique commonly used in QE studies, to determine which technique requires the least human coding to produce a sufficiently accurate classifier. To do this, we constructed multiple training sets from a large dataset used in prior QE studies and designed a Monte Carlo simulation to test the performance of the two techniques systematically. Our results show that nCoder can achieve high predictive accuracy with significantly less human-coded data than a neural network.

Keywords: Coding, Automated Classifiers, Machine Learning, Active Learning, nCoder.

1 Introduction

Coding is a process of defining concepts of interest (Codes) and identifying where they occur in qualitative data [23]. Broadly speaking, coding can be accomplished in two ways: (a) a human can read each segment of data and decide whether or not a given Code is present (or to what extent it is present); or (b) a computer can apply a classification algorithm that takes each segment of data as input and returns a coding decision as output. Because many quantitative ethnography (QE) studies involve more data than a human could reasonably read—let alone code—researchers often use automated classifiers to code their data. In order to do that, however, QE researchers need to develop classification algorithms that reliably predict the coding decisions of a human rater. Of course, developing an automated classifier typically requires some amount of human-coded data, which raises a key question: How much data does a human need to code in order to produce an accurate classification algorithm?

QE researchers typically use machine learning (ML) techniques to develop classification systems [6, 18]. Most commonly, they use either (a) traditional supervised learning techniques—such as support vector machines or neural networks—which induce a classifier from a set of human-rated excerpts (linear process), or (b) active learning techniques, in which the machine can repeatedly query a human rater to induce a classifier (iterative process).

However, it is impossible to determine *a priori* the size of the sample needed to train a classifier using any supervised learning technique, so comparison of machine learning techniques can only be done empirically [8]. To do this, researchers typically train classifiers on the same training data using traditional supervised learning techniques—and in some cases, active learning approaches (see, e.g., [4, 11]). While this makes it possible to compare the performance of classifiers on a given sample, it does not control for sampling bias, and thus generalizability is limited. Furthermore, and most importantly, these studies typically do not reflect the non-deterministic nature of the human-in-the-loop active learning process. Because the outcomes of active learning processes are contingent on the specific interactions between the machine and the human, conducting an experiment to examine this requires a significant investment of human labor. Hence, to the best of our knowledge, there have been no systematic comparisons between traditional supervised learning and active learning techniques, and thus no basis for determining the amount of human coding required to train a classifier with sufficient predictive accuracy.

In this paper, we systematically compare the performance of a traditional supervised learning approach and an active learning approach on a given dataset. Specifically, we compare an artificial neural network, a powerful traditional supervised learning technique, with nCoder, an active learning technique commonly used in QE studies. In a pilot study using one large dataset to develop automated classifiers for two Codes, we found that nCoder can achieve high predictive accuracy with significantly less human-coded data than a neural network.

2 Theory

2.1 Coding

At the most basic level, a Code is a label that is applied to some segment of data. But each time we apply a label to a piece of data, we are making an assertion about *meaning*. In the context of QE research, coding decisions create a linkage between some record of *events* (data) and *interpretation* of those events, reflecting the linkage asserted by Codes between the cultural meanings that people in some community construct and researchers' theoretically grounded interpretations of that culture [23].

Because coding is the lowest level at which assertions about the meaning of data are made, it is particularly important that coding decisions are accurate interpretations—that is, interpretations that both members of the community being studied and informed researchers would agree are fair representations of the community, of theory, and of the data itself. If we lack sufficient certainty that the codes in data mean what we think they mean, then any analysis conducted on those codes will be at best useless and at worst epistemically violent.

To achieve the high level of coding accuracy that this requires, researchers traditionally code data by hand. While there are many approaches to manual coding [15, 17, 21], all of them involve reading segments of data and deciding whether or not a code is present (or to what extent it is present). That is, the researcher makes each assertion about the interpretation of a segment of data based on close reading and expertise. But this approach is also slow, laborious, and prone to random error [19], and many communities now produce records of events that are far too large to code in this way.

As a result, researchers often use *automated classifiers* to code data at scale. Automated classifiers are algorithms, or sets of rules, that take features of data as input and return coding decisions as output. In the case of text data, the rules could be simple—for example, if the word "warbler" is present, the Code BIRD is applied—or they could involve complex interactions of words, grammar, syntax, capitalization, punctuation, sentence length or other features of written language. But whatever their rules are, automated classifiers are evaluated based on how well they *predict* the coding decisions of a human expert—that is, they are evaluated based on some measure of *inter-rater reliability* (IRR) between the classifier and a human expert [9]. Because codes are assertions of meaning, automated classifiers need to make the coding decisions that a human *would have* made if the human had read and coded the data.

To develop or *train* automated classifiers that achieve high predictive accuracy relative to human experts—that is, that meet or exceed some standard for IRR—researchers typically use one or more techniques broadly classed as *machine learning*.

2.2 Machine Learning

To develop automated classifiers for text data, machine learning techniques analyze corpora and combine features of text into classification rules [10]. These techniques can be broadly divided into (a) unsupervised learning and (b) supervised learning. *Unsupervised learning* techniques, such as those broadly classed as *topic modeling* (e.g., latent Dirichlet allocation [16]), consist of generative models that take a set of raw, uncoded text data and extract groups of keywords that are related to one another. These groups can be taken as codes, and human researchers can inspect the keywords (i.e., the classification rules) to try to discern the Codes to which they refer. Unsupervised models are often used in exploratory analyses, as they work quickly and require no initial coding—that is, they produce candidate automated classifiers with virtually no human effort. However, the resulting codes may not map to Codes grounded in theory, nor to meaningful elements of a community's culture. Moreover, codes generated by unsupervised learning techniques often achieve poor predictive accuracy [2, 6, 23].

To induce automated classifiers that better align with theory and the culture of the community being studied, researchers more commonly use *supervised learning* techniques, which operate not on raw text data but on coded text data. That is, supervised learning techniques are given information about which segments of text are associated with some Code and which are not. There are many supervised learning techniques, but they are broadly classed into *traditional supervised learning* and *active learning*. Traditional supervised learning techniques—including support vector machines, naïve Bayes, logistic regression, and neural networks—use a single set of human-coded data as an input to induce a model that predicts coding decisions on an unseen dataset. Active learning techniques—including Prodigy and nCoder—facilitate multiple cycles involving interactions between the machine and the human rater to guide incremental classifier refinement [22, 24, 25]. That is, where traditional supervised learning is linear, active learning is iterative.

However, all supervised learning techniques present several challenges for researchers. While such techniques generally perform better when the amount of human-coded data used to train the classifier is larger [20], it is not possible to compute *a priori* the size of the sample needed to train a classifier that achieves sufficient predictive accuracy. This is because the size of the sample needed to train a classifier is dependent in part on properties of that classifier (or its corresponding Code), which are difficult if not impossible to determine in advance. For example, a Code that appears infrequently in data may require more training data than one that appears frequently, and the frequency, or *base rate*, of a code in data can only be estimated once a human rater has coded some data. Thus, it is difficult to determine whether different techniques for developing automated classifiers differ significantly in the amount of human coding they require.

2.3 Assessing the Performance of Machine Learning Approaches

Research on the comparative advantages of different machine learning techniques has thus far been limited to (a) multi-case comparisons of traditional supervised learning approaches or (b) single-case studies of active learning approaches. In these studies, researchers compare traditional supervised learning techniques to one another using the same set of human-coded training data, and they compare traditional supervised learning with active learning by giving the active learning algorithm the same training data but uncoded. In such studies, the predictive accuracy of each machine learning technique is based on some IRR metric between the classifier and a human expert, and IRR scores can be compared directly because each technique received the exact same training data. For instance, Hartmann and colleagues [12] compared ten automated text classification algorithms—including support vector machines, random forest, and naïve Bayes—across 41 social media datasets, concluding that either random forest or naïve Bayes shows the highest accuracy values across the tasks. Goudjil and colleagues [11] compared support vector machines against a novel active learning method that supplemented the SVM algorithm, concluding that the active learning method can significantly reduce the amount of human inputs while enhancing predictive accuracy.

However, existing approaches for comparing machine learning techniques do not enable *systematic* comparison due to two critical limitations. (1) For a given dataset, techniques are only compared using a single training set—that is, a single sample of the dataset. While this enables performance measurement and ranking *based on that sample*, there is no control for sampling bias, and thus there is no control for error in the measurement of predictive accuracy *based on that dataset or the population from which it was drawn*. (2) This approach does not enable comparison of active learning techniques systematically, both because the need for a human in the loop is time- and labor-intensive and because the outcomes of active learning processes are dependent on the specific interactions between the machine and the human, which are non-deterministic. As a result, there are no studies that have systematically compared the performance of traditional supervised learning and active learning.

Researchers typically assess the performance of computational or statistical techniques systematically using Monte Carlo studies [13]. Monte Carlo studies use a large number of simulated datasets (or a large number of samples from a real dataset) and calculate a test statistic for each one, resulting in an empirical sampling distribution. To do this in the case of machine learning, however, requires (a) the ability to *construct large numbers of human-coded training sets* from a given dataset, and (b) the ability to *simulate a human-in-the-loop active learning process*.

In this paper, we present a systematic comparison between an artificial neural network, one of the most powerful techniques in traditional supervised learning, with nCoder, the most commonly used active learning technique in QE research. For a given Code, we use an automated classifier previously determined to have very high predictive accuracy (Cohen's $\kappa > 0.90$, Shaffer's $\rho < 0.05$) to code the entire dataset, which enables us to sample large numbers of different *coded* training sets. To simulate the nCoder active learning process, we designed an algorithm that approximates the human's input in the process, namely seeding the classifier with initial regular expressions and deciding what action(s) to take based on disagreements between the human and the classifier during training. This enables us to address the following research question: *Which machine learning technique requires the least human-coded data to train a classifier that achieves high predictive accuracy?*

3 Methods

3.1 Monte Carlo Simulation Study Design

In this pilot study, we compare two supervised learning approaches—a neural network (traditional supervised learning) and nCoder (active learning)—to determine how much human-coded data is necessary to achieve classifiers with high predictive accuracy. To do this, we used a single large dataset for which there is a set of existing automated classifiers (regular expression-based codes) that were validated by an expert human rater previously. These classifiers enable us to simulate human coding decisions at scale and also to simulate the active learning process. In what follows, we describe the general simulation design—which could be used with other datasets and other machine learning techniques—and in the subsequent sections, we describe in more detail how the Monte Carlo simulation was implemented in our study.

To test the performance of the two machine learning processes, we randomly split the dataset into two subsets: (a) a *development* set and (b) a *prediction* set. Each set was coded with the validated classifier to simulate human ratings. We used the development set to train automated classifiers, while the prediction set was used to evaluate the predictive accuracy of the classifiers.

After partitioning the data, we selected random samples of length n from the development set, where n was multiples of 100 up to 3,000 (i.e., 100, 200, 300, ⋯, 2,900, 3,000). We set 3,000 as the upper bound because it is well above the reasonable amount of data that human would typically code in order to train a classifier in QE context.

A classifier was constructed and developed for each sample. For the implementation of the neural network approach, each sample was given to the neural network algorithm. For nCoder, we designed a human-in-the-loop simulation algorithm where a classifier was developed iteratively using 100 items at a time, until a total of 3,000 lines was reached. (Detailed descriptions on the implementation of each algorithm are given below).

For each classifier, the classifier training process was repeated 100 times for each of two Codes used in this study. We used those developed classifiers to code the *prediction set* and computed Cohen's kappa against the simulated human coding decisions produced by the previously validated classifier. This allowed us to compute a confidence interval for the mean kappa values for each sample of length n for each approach and each Code.

3.2 Implementation of the neural network

Neural networks are a class of supervised learning techniques that loosely model the neurons in a biological brain. A neural network consists of (a) an input layer, a set of nodes that takes features from human-coded data; (b) an output layer, a set of nodes that produces an outcome of 0 or 1 in the case of binary classification; and (c) some number of hidden layers, sets of nodes that are in between the input and output layer that process features from the previous layer(s) and pass them to the next layer of nodes [3, 14].

There are multiple ways to implement a neural network, but for this pilot study, we used it as a traditional supervised learning technique. We implemented the neural network model that Cai et al. designed in [5]—the model uses an embedding layer that represents each unique word by a vector at the beginning, followed by one bidirectional Long Short Term Memory (LSTM) layer, and a sigmoid layer as the probability of the output, with a cutoff threshold of 0.50. That is, a probability output for a given excerpt that is greater than 0.50 is classified as 1, and otherwise 0.

For each sample, we preprocessed its text data with tokenization and lemmatization. Then, this preprocessed set, which was also coded with the pre-validated classifier to simulate a human-coded training set, was given to the neural network to develop an automated classifier. The algorithm randomly selected 20% of the given training set and used it as a validation set to tune the hyperparameters of a classifier.

3.3 Simulation of the nCoder active learning process

nCoder is an active learning technique commonly used in QE research. nCoder facilitates construction and validation of a set of regular expressions, such that any excerpt in which one or more expressions occurs is classified as 1, and otherwise 0. To develop a classifier in nCoder:

1. A human rater seeds an automated classifier with one or more regular expressions.
2. nCoder randomly selects segments of previously unseen data and presents them to the human rater to evaluate for the presence or absence of the Code.
3. nCoder computes Cohen's kappa between the human coder and the classifier for the Code.
   a. If the agreement is above the threshold (typically $\kappa > 0.90$), the classifier development process terminates, and the current set of regular expressions is used as a classifier.
   b. If the agreement is below the threshold, the machine shows the human rater the excerpts on which there is disagreement, the rater uses those disagreements to guide refinement of the classifier, and then the process is repeated beginning with Step 2 until the threshold for agreement is met (3a).

In order to model the human-in-the-loop process of nCoder, we designed the *Robocoder*, an algorithm that simulates this process. This algorithm compares the classifier being trained (which we term the *User Regex List* or URL) with a simulated human's coding decisions. Because the users who develop automated classifiers using nCoder possess expertise and deep knowledge of the data and the Code for which the classifier is being developed, we assume that the simulated human rater does not make errors in coding segments for this study. Therefore, we simulate the human's coding decisions as the classification results from a pre-validated set of regular expressions (the *Ideal Regular expression List* or IRL). To simulate the nCoder classifier development process for a given Code on a given dataset, the Robocoder uses the following process:

1. The Robocoder randomly selects five of the ten most frequent regular expressions from the IRL to seed the URL. We chose five because nCoder suggests starting with at least five regular expressions in the beginning, and we select them from the ten most frequent because human experts are likely to seed a classifier with common expressions first.
2. Each sample (length of 100) is coded with both the URL (classifier being developed) and the IRL (simulated human).
3. The *prediction set* is coded with both the URL and the IRL, and Cohen's kappa is computed between them.

4. The Robocoder identifies whether there are one or more excerpts from the sample drawn in Step 2 for which the coding decisions of the URL and IRL are different.
   a. If there are any discrepancies in the ratings, the Robocoder will resolve differences between the ratings of the URL and IRL as follows:
      i. For each item where the coding decisions of the URL and IRL are different, the Robocoder finds all the regular expressions from the IRL that are not in the URL but that would have been satisfied by the excerpt.
      ii. Among those expressions, the Robocoder selects the one that has the highest base rate and adds it to the URL, simulating how a human would expand the regular expression list in response to a false negative by the URL.
      iii. After all differences are resolved, the Robocoder returns to step 2.
   b. If there are no discrepancies between the ratings of the URL and IRL, the Robocoder checks whether the length of the accumulated sample sets is 3,000.
      i. If Yes, the Robocoder stops.
      ii. If No, the Robocoder returns to Step 2.

We repeated this process 100 times for each of the two Codes used in this study.

3.4 Data

The dataset used in this study consists of 50,888 chat utterances from an engineering virtual internship, *Nephrotex*, in which students work in teams to design a new filtration system for kidney dialysis machines [7]. We used the coding scheme developed by Arastoopour Irgens et al. [1], which includes a set of regular expression classifiers for each Code. For this study, we used two Codes: CLIENT/CONSULTANT REQUESTS and TECHNICAL CONSTRAINTS, as each has the lowest (0.07) and highest (0.16) base rate. The regular expressions validated by Arastoopour Irgens et al. [1] showed high κ values with a human expert, and thus can be used to simulate human coding decisions (See Table 1). Table 2 shows the frequency of each regular expression for each of the two codes.

Table 1. Base rate, the number of regular expressions, and the Cohen's kappa for each Code.

Code	Base Rate	Number of Regular Expressions	Agreement between Human Expert and Classifier
CLIENT/CONSULTANT REQUESTS	0.07	24	$\kappa = 0.94$
TECHNICAL CONSTRAINTS	0.16	21	$\kappa = 1$

Table 2. Regular expressions for CLIENT/CONSULTANT REQUESTS and TECHNICAL CONSTRAINTS (Base rates are sorted by the descending order and displayed up to three or four decimal point).

CLIENT/CONSULTANT REQUESTS		TECHNICAL CONSTRAINTS	
Regular Expression	**Base Rate**	**Regular Expression**	**Base Rate**
^(?:(?!\bexternal).)*\bconsultant(?!.*\bexternal)	0.034	\bmanufacturing process	0.034
\binternal consultant	0.012	\bprocesses	0.012
\bpatient	0.011	\bnano	0.011
\brequirement	0.010	\bhydro	0.010
\brequest	0.009	\bsteric	0.009
\bstandard	0.004	\bvapor deposition polymerization	0.004
\bminimum	0.004	\bphase inversion	0.004
\bPadma	0.002	\bnegative charge	0.002
\bAlan	0.002	\bPRNLT	0.002
\bRudy	0.002	\bvapor	0.002
\bWayne	0.002	\bPolyamide	0.002
\bsatisfy	0.002	\bmaterials	0.002
\bMichelle	0.002	\bchemical	0.002
\bhospital	0.002	\bdry-jet	0.001
\bDuChamp	0.001	\bjet	0.001
\bcomfort	0.001	\bbiological	0.001
\bProctor	0.001	\bcarbon nanotube	0.001
\bAnderson	0.001	\bpolysulfone	0.001
\bRao	0.001	\bPESPVP	0.001
\bHernandez	0.001	\bsurfactant	0.0008
\buser	0.001	\bPMMA	0.0005
\bclient	0.0009	\bCNT	
\bsafety	0.0008		
\brecommendations	0.0005		

4 Results

Figs. 2 and 3 show the mean Cohen's kappa values with 95% confidence intervals between the developed classifiers (URLs) and the pre-validated classifier used to simulate human coding (IRL) on the prediction set for samples of length n (See Fig.1 for the Code CLIENT/CONSULTANT REQUESTS and Fig. 2 for the Code TECHNICAL CONSTRAINTS). These plots demonstrate that nCoder requires considerably smaller sample size to train an accurate classifier than a neural network. Even with 3,000 coded excerpts as a training set, the neural network's mean kappa remained below 0.90, a common threshold used in QE research. On both of the Codes, while nCoder required 700 segments to be coded in order for the lower bounds of the confidence intervals to be above 0.9, there is no case in our plots that the lower bounds of the confidence intervals are above 0.9 for the neural network, meaning that the neural network requires coding more than 3,000 segments to obtain an accurate classifier.

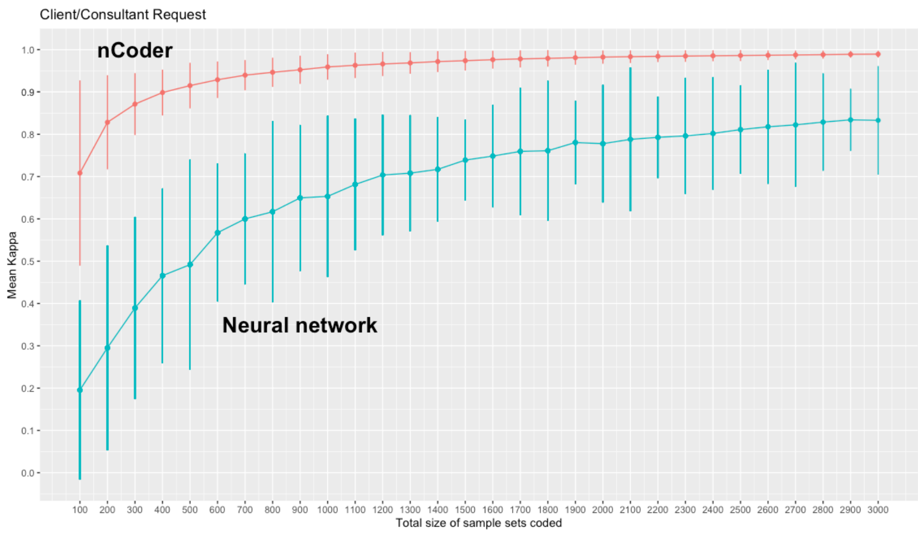

Fig. 1. Mean Cohen's kappa metrics with 95% confidence intervals between the developed classifiers (URLs) and the pre-validated classifier simulating human coding (IRL) on the prediction set for the Code CLIENT/CONSULTANT REQUESTS

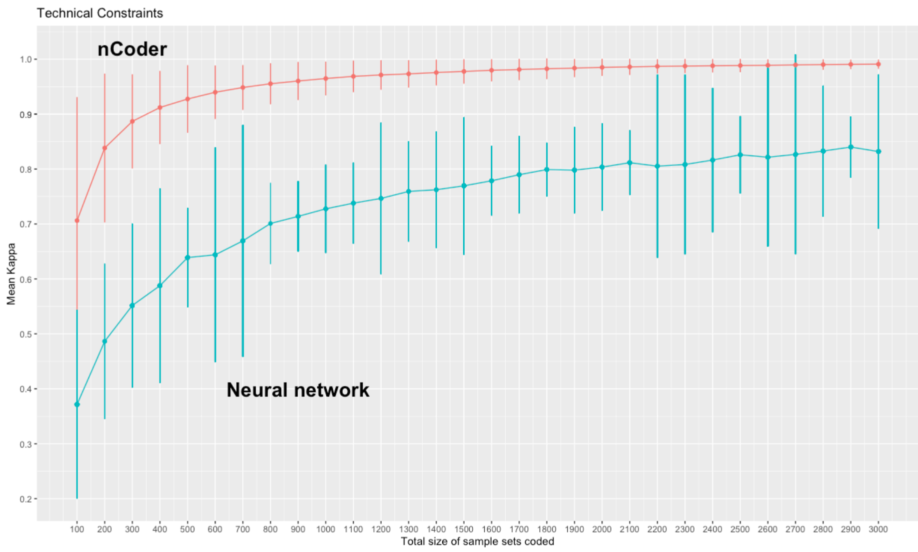

Fig. 2. Mean Cohen's kappa metrics with 95% confidence intervals between the developed classifiers (URLs) and the pre-validated classifier simulating human coding (IRL) on the prediction set for the Code TECHNICAL CONSTRAINTS

5 Discussion

In this paper, we present a comparative study that evaluates the amount of human-coded data needed by a neural network model (traditional supervised learning) and nCoder (active learning) to develop an automated classifier that achieves high accuracy. Our pilot results show that for the two Codes we tested, nCoder requires significantly less human-coded data to train a classifier that achieves sufficient predictive accuracy compared to the neural network technique.

This study enabled a systematic comparison between traditional supervised learning and active learning approaches by conducting a Monte Carlo Simulation. Using a pre-validated automated classifier to code the entire data that has very high predictive accuracy, large numbers of different coded training sets were sampled to control for sampling bias. Most importantly, we designed an algorithm that simulates nCoder, a human-in-the-loop active learning approach.

However, it is important to note that these results are largely dependent on the architectural design of machine learning approaches and the human-in-the-loop simulation. In our study, the design of the Robocoder closely resembles how the actual human user would update the classifier when resolving differences between the classifier and the human. Therefore, we can model the behaviors of the users without conducting cost and resource-intensive experiments with humans. However, we hypothesized that human raters do not make any mistakes or errors in coding process or when refining the classifier. For instance, if the human-coded inputs have a substantial amount of error in the beginning, while nCoder can correct the errors through iterations between the human and the machine, the neural network model would have low predictive performance, because it is a linear process that cannot rule out any errors once the process begins. On the other hand, if a human rater keeps adding incorrect regular expressions that they think are important, it will require more iterations, so the number of samples that human needed to code will increase in nCoder. Hence, we expect to address these issues in the future studies.

Yet one important contribution of this study is that the presented method can be expanded to any other Codes, datasets, and/or machine learning algorithms. By randomly generating large numbers of coded training sets from a given dataset, we can systematically control for the sampling bias that the previous studies using a single training set did not address. In addition, the ability to simulate a human-in-the-loop active learning process enables more thorough measurement by reflecting the iterative and complex nature of the human involvement process.

Acknowledgements. This work was funded in part by the National Science Foundation (DRL-1661036, DRL-1713110, DRL-2100320), the Wisconsin Alumni Research Foundation, and the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin-Madison. The opinions, findings, and conclusions do not reflect the views of the funding agencies, cooperating institutions, or other individuals.

6 References

1. Arastoopour, G. et al.: Teaching and assessing engineering design thinking with virtual internships and epistemic network analysis. Int. J. Eng. Educ. 32, 3, 1492–1501 (2016).
2. Bakharia, A.: On the equivalence of inductive content analysis and topic modeling. In: International Conference on Quantitative Ethnography. pp. 291–298 Springer (2019).
3. Baradwaj, B.K., Pal, S.: Mining educational data to analyze students' performance. ArXiv Prepr. ArXiv12013417. (2012).
4. Bull, L. et al.: Active learning for semi-supervised structural health monitoring. J. Sound Vib. 437, 373–388 (2018).
5. Cai, Z. et al.: Neural recall network: A neural network solution to low recall problem in regex-based qualitative coding. In: Proceedings of the 15th International Conference on Educational Data Mining. p. XX (2022).
6. Cai, Z. et al.: Using Topic Modeling for Code Discovery in Large Scale Text Data. In: International Conference on Quantitative Ethnography. pp. 18–31 Springer (2021).
7. Chesler, N.C. et al.: A Novel Paradigm for Engineering Education: Virtual Internships With Individualized Mentoring and Assessment of Engineering Thinking. J. Biomech. Eng. 137, 2, 024701 (2015). https://doi.org/10.1115/1.4029235.
8. Cho, J. et al.: How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? ArXiv Prepr. ArXiv151106348. (2015).
9. Eagan, B.R. et al.: Can We Rely on IRR? Testing the Assumptions of Inter-Rater Reliability. 4 (2017).
10. González-Carvajal, S., Garrido-Merchán, E.C.: Comparing BERT against traditional machine learning text classification, http://arxiv.org/abs/2005.13012, (2021).
11. Goudjil, M. et al.: A Novel Active Learning Method Using SVM for Text Classification. Int. J. Autom. Comput. 15, 3, 290–298 (2018). https://doi.org/10.1007/s11633-015-0912-z.
12. Hartmann, J. et al.: Comparing automated text classification methods. Int. J. Res. Mark. 36, 1, 20–38 (2019).
13. Harwell, M.R.: Summarizing Monte Carlo results in methodological research. J. Educ. Stat. 17, 4, 297–313 (1992).
14. Hernández-Blanco, A. et al.: A systematic review of deep learning approaches to educational data mining. Complexity. 2019, (2019).
15. Holton, J.A.: The coding process and its challenges. Sage Handb. Grounded Theory. 3, 265–289 (2007).
16. Jelodar, H. et al.: Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. Multimed. Tools Appl. 78, 11, 15169–15211 (2019).
17. Khandkar, S.H.: Open coding. Univ. Calg. 23, 2009 (2009).

18. Larson, S. et al.: Healthcare professionals' perceptions of telehealth: analysis of tweets from pre-and during the COVID-19 pandemic. In: International Conference on Quantitative Ethnography. pp. 390–405 Springer (2021).
19. Miles, M.B., Huberman, A.M.: Qualitative data analysis: An expanded sourcebook. sage (1994).
20. Ramezan, C.A. et al.: Effects of training set size on supervised machine-learning land-cover classification of large-area high-resolution remotely sensed data. Remote Sens. 13, 3, 368 (2021).
21. Scott, C., Medaugh, M.: Axial coding. Int. Encycl. Commun. Res. Methods. 10, 9781118901731 (2017).
22. Settles, B.: Active Learning Literature Survey. 47.
23. Shaffer, D.W., Ruis, A.R.: How We Code. In: Ruis, A.R. and Lee, S.B. (eds.) Advances in Quantitative Ethnography. pp. 62–77 Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-67788-6_5.
24. Yu, D. et al.: Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion. Comput. Speech Lang. 24, 3, 433–444 (2010). https://doi.org/10.1016/j.csl.2009.03.004.
25. Prodigy · An annotation tool for AI, Machine Learning & NLP, https://prodi.gy, last accessed 2022/05/23.