Editors' Suggestion

Method to assess the trustworthiness of machine coding at scale

Rebeckah K. Fussell, Emily M. Stump, and N. G. Holmes

Laboratory of Atomic and Solid State Physics, Cornell University, Ithaca, New York 14853, USA

(Received 8 November 2023; accepted 1 February 2024; published 6 March 2024)

Physics education researchers are interested in using the tools of machine learning and natural language processing to make quantitative claims from natural language and text data, such as open-ended responses to survey questions. The aspiration is that this form of machine coding may be more efficient and consistent than human coding, allowing much larger and broader datasets to be analyzed than is practical with human coders. Existing work that uses these tools, however, does not investigate norms that allow for trustworthy quantitative claims without full reliance on cross-checking with human coding, which defeats the purpose of using these automated tools. Here we propose a four-part method for making such claims with supervised natural language processing: evaluating a trained model, calculating statistical uncertainty, calculating systematic uncertainty from the trained algorithm, and calculating systematic uncertainty from novel data sources. We provide evidence for this method using data from two distinct short response survey questions with two distinct coding schemes. We also provide a real-world example of using these practices to machine code a dataset unseen by human coders. We offer recommendations to guide physics education researchers who may use machine-coding methods in the future.

DOI: 10.1103/PhysRevPhysEducRes.20.010113

I. INTRODUCTION

Education researchers are exploring the use of machine learning to automate the process of applying coding schemes to students' written work. Typically, human coders painstakingly apply coding schemes to such responses. "Machine coding," by contrast, promises to utilize machine learning and natural language processing (NLP) tools to dramatically increase the efficiency of coding new data. With efficient machine coding, coding schemes could be applied at scale: across years, courses, and institutions. This scale would allow researchers to answer research questions and to evaluate diverse populations of students [1] in ways that have been unavailable in the past because of the large amounts of manual coding that would be required. In addition to efficiency improvement, machine coding could improve consistency. Machine coding algorithms can be fixed such that they use the same procedure to code each response, whereas a human coder might code responses inconsistently because of fatigue or lack of clarity about the rules of the coding scheme.

Skeptics of machine learning, however, worry that the algorithms will introduce or perpetuate biases in training data [2] or distrust that an imperfect algorithm may be preferable to imperfect human judgment [3]. Here, we seek to address this skepticism by presenting methods of evaluating the trustworthiness of a machine learning algorithm specific to physics education research contexts, drawing on data analysis techniques common to experimental physics (namely, quantifying statistical and systematic uncertainties).

We argue that these techniques particularly support physics education researchers because we cannot necessarily rely on mainstream machine learning techniques. For instance, one common NLP exercise is to train and test algorithms using aggregated banks of news headlines [4,5], which contain many thousands of unique instances of written text and come prelabeled by news curators with codes like "politics" and "wellness." By contrast, education datasets of student writing are often small (e.g., one thousand or fewer short paragraph responses) and training data generally would need to be labeled by researchers.

In this work, we focus on machine learning processes that can be designed to mimic human coding of students' written text (responses). These processes center around the coder learning a set of rules (a "trained model" in machine learning or a "coding scheme" in human coding) that define whether or not a label should be applied to a type of response. The coder then applies labels to responses based on these rules. Supervised machine learning algorithms are particularly aligned to mimic human coding. A supervised algorithm uses a training set of human-coded data to learn the set of rules, then applies these rules to machine code any new data shown to the trained algorithm.

Machine coding with a supervised algorithm in this manner is an emerging research practice in physics education research, not simply a technical procedure. Machine coding is a practice that aims to make quantitative measurements and claims about student responses, such as measuring the frequency of categories of ideas and themes in students' written responses. We contrast these quantitative measurements and claims to qualitative measurements and claims, such as identifying the existence of ideas and themes. As a new quantitative research practice, it is vital in these early stages to establish high-quality normative practices [6].

Unfortunately, physics education research has not yet established these norms, such that we can apply machine coding to student text at scale, which we relate to two key issues. First, relatively few studies have focused on using machine learning for making quantitative claims (for example, claims about the quantity or frequency of codes within a dataset). Instead, most of the existing work by education researchers that utilizes machine learning of written text focuses on making qualitative claims. For example, researchers have used unsupervised techniques to identify themes within text and video data and to generate insights that complement human analytical insights [7–11]. Others have performed computational grounded theory to aid the human analyst in developing theoretical constructs from text data [12–14].

Second, the studies that provide "proof of concept" towards automating coding to make quantitative claims at scale [15–24] primarily rely on human-coded data to test the validity of their models and to establish trust in any quantitative claims. For example, researchers using thematic analysis (sometimes called topic modeling in the natural language processing field) to evaluate the prevalence of various themes within text data [25–28] noted that, due to the unsupervised nature of their form of analysis, these prevalence values cannot be taken to be a measurement of the ground truth as there is no way to assess the accuracy without comparing to human coders [25]. Researchers using supervised algorithms commonly evaluate the validity of machine coding by computing reliability metrics that compare the codes assigned by the algorithm to the codes assigned by a human. Common reliability metrics include Cohen's kappa, quadratic weighted kappa, accuracy, recall, precision, F1 score, and area under the receiver operating characteristics curve (AUC—ROC) [12,15–18]. When calculating these reliability metrics, researchers need a large human-coded dataset that can be split into sufficiently large training and test sets.

Even then, while researchers have developed algorithms that can surpass threshold values for reliability [e.g., [12,15,17,21]], it is not clear what to make of these reliability measures when it comes time to make quantitative education claims about machine coded datasets. For example, if a machine coding model obtains a Cohen's kappa value of 0.65, indicating "substantial agreement" with a human coder [29], we are left with questions such as, did the computer systematically over- or underestimate the code relative to the human coder? If we make a quantitative education claim using the machine coded data, how much uncertainty should we maintain toward that claim given this value of Cohen's kappa?

Furthermore, there is no guarantee that the threshold will continue to be met in other, novel datasets. The field does not yet have established norms for evaluating the validity of the coding in novel datasets. This concern is especially relevant when the datasets being assessed are small or systematically different from the training dataset, as in most human-coded education datasets. For example, the reliability metric may change significantly if the algorithm were applied to a test set from a new institution or student population. Researchers in computer science and statistics are developing advanced methods to tackle this problem by improving the ability of trained models to accurately process test data that is unlike the training data using a causal inference perspective [30] or by leveraging the causal instincts of human annotators [31]. These methods, however, require substantial resources (e.g., amount of data and amount of human coding) beyond what is accessible to physics education researchers.

For now, as physics education researchers rely on the reliability metric approach to evaluating machine coding, we are left with a dilemma: if the only way to evaluate the accuracy and reliability of machine coding is to human code all (or at least a very large fraction) of the data, then what is the benefit of machine coding?

Here we propose a set of methodological practices that can be used to evaluate the trustworthiness (accuracy and reliability) of machine coded data to make quantitative education research claims while minimizing the need for human coding through uncertainty quantification. Researchers in other fields have similarly analyzed the quantification of uncertainty when using natural language processing [32], but these methods do not provide specific steps that allow researchers in our field to make and evaluate trustworthy quantitative claims. Our goal is to contribute to the conversation on methodological standards and review of quantitative claims within PER [33,34], and we encourage others in the community to expand on our framework.

We describe a four-part methodology for evaluating machine coding that casts machine coding results as experimental measurements with associated uncertainties (statistical and systematic), summarized in Fig. 1. The methodology (i) evaluates a trained model, (ii) quantifies statistical uncertainty, (iii) quantifies systematic uncertainty from the trained algorithm, and (iv) quantifies systematic uncertainty in novel datasets. We argue this approach addresses the concerns described above and incorporates best practices in expressing measurements with uncertainty [35].

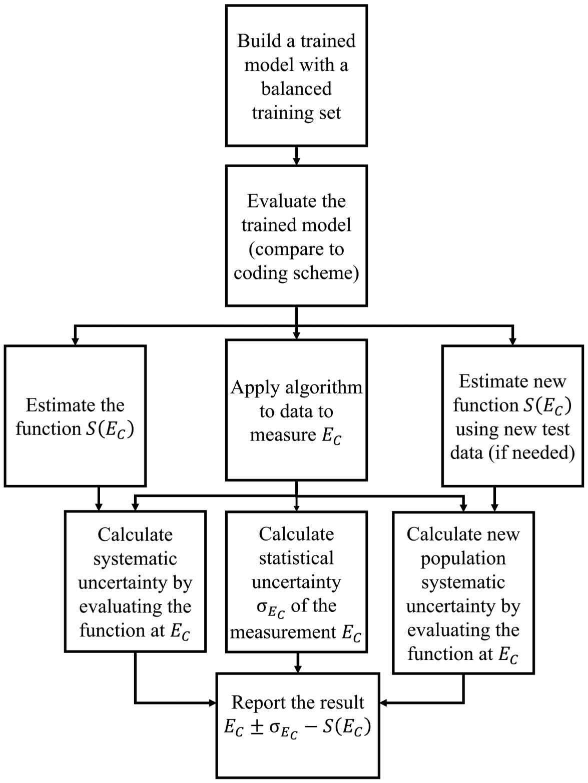
Build a trained model with a balanced training set
Evaluate the trained model (compare to coding scheme)
Estimate the function $S(E_C)$
Apply algorithm to data to measure E_C
Estimate new function $S(E_C)$ using new test data (if needed)
Calculate systematic uncertainty by evaluating the function at E_C
Calculate statistical uncertainty σ_{E_C} of the measurement E_C
Calculate new population systematic uncertainty by evaluating the function at E_C
Report the result $E_C \pm \sigma_{E_C} - S(E_C)$

FIG. 1. Flowchart depicting the stages of the methodology.

The rest of this paper is organized as follows. First, in the methods section we present our data sources and our approach to training a supervised machine learning algorithm that uses natural language processing to perform machine coding. Then, we devote a section to each part of the methodology. We then work through an example of applying this methodology to a dataset we have not coded by hand. Lastly, we discuss the limitations of the evidence available to us so far and suggest opportunities for future work.

II. METHODS

A. Data sources

In this work, we use two distinct sets of student responses to open-response survey questions: the *trustworthy* data (collected using a survey question from [36]) and the *sources* data (from [37]).

The *trustworthy* data consists of 1958 responses to the survey question "How do you know whether or not an experimental result is acceptable or trustworthy? What gives you confidence that the data is trustworthy?" from Ref. [36]. The question was included on pre- and post-surveys administered at Cornell University across three different academic years. Data are from students taking calculus-based introductory mechanics and electricity and magnetism courses. We developed a coding scheme with seven codes adapted from the scheme used in Ref. [36]: consistent results, uncertainty, expected result, good methods, ethics, peer review, and statistics. In this coding scheme, individual responses can receive multiple codes and the coders identify the presence or absence of each code one at a time.

In the main analysis below, we focus on one of these codes: consistent results (CR). For the CR code, two human coders achieved a Cohen's kappa value of 0.9 in 10% of the data. A kappa value of 0.8–1.0 is considered to be very good agreement [29]. Then one human coder hand-coded 1672 of the responses—all the data from semesters prior to Fall 2022. The remaining 286 responses were from the end of the Fall 2022 semester and were not coded by humans.

The CR code is defined as "the same result is obtained when looking at more data, either through repeated trials or comparing with other people." The inclusion criteria for the CR code are

1. repeated, repeatable, and/or consistent across independent measurements,
2. experiment conducted numerous times,
3. obtaining consistent results,
4. repeating experiments with different methods,
5. collecting additional data to verify trends,
6. multiple trials,
7. low variance or deviation from mean,
8. large sample size,
9. comparing with peers, other research groups, different scientists, published results,
10. replicability or reproducibility,
11. consistency of results across different research groups,
12. others get the same results.

In Sec. VII, we also examine another of these codes: the uncertainty code, which will be defined there.

The *sources* data consist of 2413 responses to a survey where respondents were shown experimental physics scenarios and fictional distributions of data. The responses in this analysis are to the question "What is causing the shape of the distribution measured by the students? List as many causes as you can think of." from Ref. [37]. The survey was administered in over a dozen courses at 12 institutions. In total, the responses were written by 753 students, each of whom wrote multiple sources for each experimental scenario (each source is treated as a unique response). Upper-division students were shown up to four different experiments and prompted to list sources of uncertainty for each, so the dataset consists of students' ideas about uncertainty for projectile motion (468 responses), Brownian motion (290 responses), single slit (239 responses), and Stern-Gerlach (282 responses) experiments [37]. An additional 1134 responses for the projectile motion scenario came from introductory-level students [38]. The responses were coded based on three categories—limitations, principles, and other—as described in Ref. [37]. Two human coders achieved a Cohen's kappa of 0.85 across 40 responses (10 responses from each experiment). Then each coder coded a portion of the rest of the data.

In the analysis below, we focus on the limitations code (L), which was by far the most frequent code. The L code is defined as: “A distribution is caused by practical limitations in an experiment owing to our inability to perfectly model and measure a real-world system.” No inclusion criteria are listed in the coding scheme. Instead, human coders primarily used exclusion criteria in coding. The exclusion criteria for the L code are

1. A distribution is caused by some principle of theoretical physics (theoretical abstraction of an experiment, e.g., quantum theory) or of experimentation (measurement is inherently random or has an inherent normal distribution).
2. Vaguely worded responses about “uncertainty” or “random errors.”
3. Physical mechanisms that determine the position (distance) of the central value (average value) but are not varying between experimental trials, e.g., gravity.

We chose to use these two codes because they are reliably measured by human coders and provide one example of a code mostly defined by inclusion criteria (CR) and one example of a code mostly defined by exclusion criteria (L). This choice allows us to investigate the applicability of these methods across a range of rule types that physics education researchers may encounter.

B. Natural language processing

In developing the machine coding algorithms, we used a one-vs-all (OVA) approach, where we built separate algorithms that focused on one code at a time. The machine applies a label of 0 or 1 if the code is absent or present, respectively. The OVA approach mirrors the decision-making process the human coders used (namely, focusing on one code at a time and reading each response for evidence of that code) and has been shown to perform as well as more complex, multicode approaches [39].

To prepare the responses for automated coding, we used the following bag-of-words natural language processing (NLP) protocol: (i) split all words in each response into individual words (often called tokens or features), (ii) fix contractions (for example, “you’re” becomes “you are”), (iii) use the Word Net Lemmatizer from the Natural Language Toolkit python package [40] to combine words from the same family such as plurals and verb conjugates, and (iv) remove any remaining whitespace, punctuation, and numbers. We then encode the modified responses as a matrix where each row corresponds to a single response and each column represents a single token (i.e., each unique word in the total set of modified responses). Each entry in the matrix is a 0 or 1 indicating whether that token is present in the response. We found in initial testing that this 0 or 1 method outperforms other common word scoring methods such as raw count of each word or term frequency inverse document frequency [41,42].

We did not filter out stop words such as “and,” “it,” and “the.” This form of filtering can be useful to reduce the size of a large dataset or to perform machine coding tasks such as categorizing technical texts [43]. In an education research context, however, we are dealing with small datasets where reducing the size of the data in pre-processing may be a less important consideration. Furthermore, stop words like “only,” “but,” and “just” may be critical when analyzing the nuances of student thinking as revealed by their responses [44].

In this paper, all machine learning models were built with bag-of-words logistic regression algorithms using the scikit-learn package in python [45]. In initial testing, we examined bag-of-words models built with logistic regression, naive Bayes, support vector machines, random forest, k-nearest neighbors, and neural networks (both a single-layer perceptron and a convolutional neural network utilizing the GloVe word embedding [46,47]). We found that logistic regression and single-layer perceptron had the highest, most consistent performance across four metrics: accuracy, precision, recall, and Cohen’s kappa. We chose to use logistic regression because it is easier to view the coefficients assigned to different features and because it has been used in other physics education research literature, such as [15].

C. Training and test sets

We split the available responses to create a training dataset and a test dataset (Fig. 2); details about the test data are further described in Sec. II D. In the majority of our analyses, the training data include $N_{\text{train}} = 600$ responses to train each algorithm. While algorithm performance generally increases as the size of the training set increases, we previously found that performance plateaus around $N_{\text{train}} = 600$ student responses from the *trustworthy* data [17]. In a few analyses, we use a smaller training set because we need additional data for testing. We note these instances as appropriate. We define p_{train} as the proportion of responses in the training set containing the particular code, where $0 \leq p_{\text{train}} \leq 1$.

During model development, we keep the training data and test data siloed from each other as suggested by Aiken *et al.* [48]. Overestimation of model performance can occur when training and test data are repeatedly split at random from the full dataset and not kept strictly separate during model development. This contamination can occur even if test data are not directly included in the training set (for instance, information from the test set can enter training during feature selection or hyperparameter optimization), especially when sample sizes are small [49]. By keeping the two sets siloed, we remove this risk.

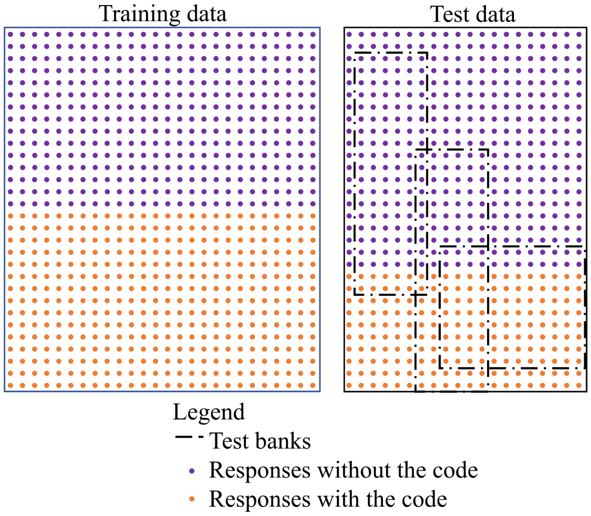

FIG. 2. The available data are split into training data and test data. Purple points represent responses without the code and orange points represent responses with the code. In this representation, three test banks of fixed size N_{bank} are drawn outlined in a dashed line, each with a different fixed proportion of responses that contain the code: 0.2, 0.5, and 0.8. Test set samples of size n are drawn at random from the test banks (in the figure, data are sorted so random samples are not pictured).

Whereas, in previous work, researchers reported results averaged across multiple training and test sets [15–17,50], here we analyze only one training set at a time, though we use multiple sample test sets (each drawn from a test bank). We suggest that this process better mimics the intended process for education research in which one trained algorithm would be used to machine code new data. In addition, with limited resources, a researcher should prioritize human coding one large training set as opposed to many smaller training sets, because a larger training set tends to create a more reliable algorithm [17].

D. Test set sampling

We divide the test data into a set of test banks, each of which is of size N_{bank}. In the analysis below, we intentionally create test banks that span the range of possible proportions of responses that contain the code (as determined by human coders) between 0 and 1 inclusive, usually a multiple of 0.1. We then draw multiple sample test sets of size n from each test bank upon which we perform our analyses. Naturally, $N_{\text{bank}} > n$ and we seek to select values of n such that N_{bank} is 2–3 times larger than n. This number allows us to balance the need to produce samples that are different from each other with the constraints of limited data.

The trained model computes the proportion of responses in the sample test set that contain the code, which we call E_C (the “computer’s estimate”). We resample the test bank a set number of times (usually 100 or 1000) to develop a distribution of E_C and to calculate $\overline{E_C}$, the mean of the distribution of repeated measurements of E_C made with the different sample test sets.

We define E_H as the proportion of responses in *each sample test set* containing the code as determined by the human coder. With enough sample test sets pulled from the test bank (that is, when the number of resamplings of the test bank is sufficiently large), the mean of E_H across the sample test sets, $\overline{E_H}$, is approximately equal to the proportion of responses in the test bank containing the code, based on statistical properties of sampling from a population.

E. Optimizing a training model

A machine coding algorithm can be optimized in multiple ways during development and researchers must make informed decisions about optimization that are relevant for their particular needs. For discussion of optimizing hyperparameters (fixed parameters that determine how the algorithm behaves) in logistic regression models, we refer the reader to [15].

In this article, we focus discussion about optimization on a practical consideration that is particularly relevant to physics education researchers, namely using training data that has an equal distribution of responses across each of the possible outcomes (referred to as “outcome balance”). For example, when using one-vs-all coding as in this work, a balanced training set would have 50% of the responses in the training dataset with the code (i.e., $p_{\text{train}} = 0.5$). We focus on this consideration as one particularly relevant to education researchers who are constrained by the size of educational datasets and the necessary investment of human coding for developing training sets.

This recommendation is motivated by previous work, such as Refs. [17,50], which suggest that an algorithm becomes less reliable when trained with a dataset with unbalanced outcomes (that is, a code is present in a majority or minority of responses). The benefits of balanced representation of the presence and absence of a code also aligns with models of human learning, such as through contrasting cases [51] and negative instances or nonexamples [52].

We demonstrate the impact of balancing the training set by training our algorithm with nine different training sets for our CR code, each with a different value of p_{train} between 0.1 and 0.9 inclusive. The size of the training set for each algorithm was 600 responses. We then create a set of test banks where each test bank is of size $N_{\text{bank}} = 200$ responses, each test bank is sampled 100 times, and each sample test set is of size $n = 100$ responses. For each algorithm, we compute the proportion of responses with the code according to the computer coder, E_C, and according to the human coder, E_H. We fix the proportion of responses in the test bank containing the code (according to human coders) to be 0.5.

We observe that the magnitude of the difference between E_C and E_H deviates from zero as p_{train} deviates from 0.5

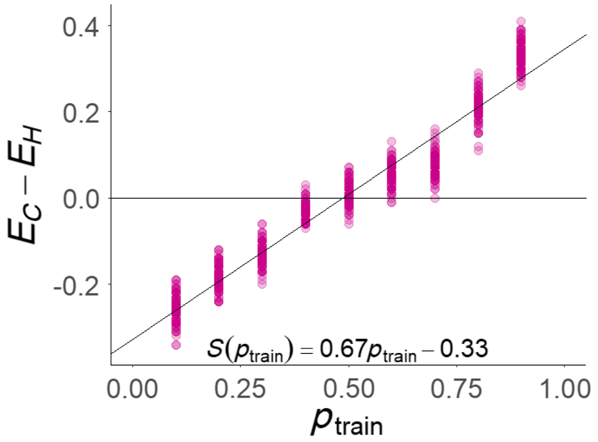

FIG. 3. The difference between the human (E_H) and computer's (E_C) estimate of the presence of a code in a test set increases as p_{train} is less balanced (i.e., far from $p_{\text{train}} = 0.5$). (Code: CR, proportion of responses in the test bank containing the code is 0.5).

(Fig. 3). The difference is such that the machine coding overestimates the prevalence of the code in the test set (i.e., $E_C > E_H$) when $p_{\text{train}} > 0.5$ and the machine coding underestimates the code (i.e., $E_C < E_H$) when $p_{\text{train}} < 0.5$, with the effect increasing as p_{train} becomes more unbalanced (farther from 0.5). Crucially, we find no offset between the machine coding and the human coding when $p_{\text{train}} = 0.5$. In Appendix A, we show that these findings also hold for the L code from the *sources* dataset (Fig. 8).

An important question is whether the lack of offset at $p_{\text{train}} = 0.5$ is based on a core principle of learning or simply that the offset is zero if p_{train} equals the proportion of responses in the test bank (which in this case was set to 0.5). We test this effect empirically in Appendix A for two different test set proportions. We indeed find that the algorithm better matches human coding when the proportion of responses with the code in the test set matches the value of p_{train}. In educational research settings, however, our goal is to apply a trained model to new data that has not been coded by humans, such that the proportion of responses containing the code is unknown. The analyses in Appendix A indicate that using training data with $p_{\text{train}} = 0.5$ is the best option to use across a range of possible test set proportions and, thus, is the best option for applying training models to new data (Fig. 9).

Thus, we propose balancing the training set as a key consideration when optimizing a trained model and we use balanced training sets throughout this paper. This consideration places constraints on the idea in previous work that increasing the size of a training set will improve the model [17]. A smaller, balanced training set may be better than a larger, unbalanced training set. We generally recommend that some balancing method (whether it be balanced training sets or a different balancing procedure such as random over-sampling of the minority case [53]) be used in future PER research involving machine coding.

One consideration for many researchers is do I have enough human-coded data to do this type of analysis? We use 600 human-coded responses in the training set, as justified above. To balance the 600 responses in the training set, it is generally necessary for human coders to process more than 600 responses in order to find 300 responses that contain the code and 300 responses that do not. In some cases, if the code is well captured by the trained algorithm, a smaller training set may be suitable. An example of such a case can be seen in Sec. VII.

The methods described up to this point leave us with a single trained model—a "machine coder"—that has learned a set of rules to consistently apply to each response. For more methodological details, see the code for this research on GitHub [54].

III. EVALUATING A TRAINED MODEL

The first step in establishing the trustworthiness of an algorithm is to check whether the trained algorithm has conceptual or theoretical validity. This is typically carried out by evaluating the outputs of the algorithm on the training set to see how it uses and connects individual words to codes [7,13]. We can also peek at the coefficients the model applies to individual words or other features. We think of this as "looking under the hood" of the algorithm to understand how it is characterizing words and codes.

A. Explanation of method

After optimizing the algorithm with a balanced training set, researchers can evaluate, qualitatively, which linguistic features the trained model uses to decide whether to include or exclude a response from a code. We identify these linguistic markers by evaluating the coefficients the model uses to process individual words. These coefficients are provided through the algorithm's logistic regression model, which models the probability p that a response contains the code as a sigmoid function of the features, x_i (in this case individual words), defined as

$$p = \frac{1}{1 + e^{-z}}, \tag{1}$$

where $z = \sum_i \beta_i x_i$ and β_i are the coefficients that best fit the training data. For each response, $x_i = 1$ if the word is present and $x_i = 0$ if the word is absent. In the machine learning process, a regularization procedure is applied that shrinks the coefficients β_i proportional to the square of the coefficient, making this a ridge regression. We denote the new coefficients (after shrinking) as κ_i. The shrinking procedure protects against overfitting by preventing the coefficients from growing too large. Note that the features x_i in this case are individual words from the training set.

The coefficient κ_i gives us information about if the algorithm uses that word as an inclusion or exclusion criteria for characterizing the code. A word with a coefficient greater than zero indicates the machine coding interprets the use of that word as evidence for *inclusion* of the response in the code, while a word with a coefficient less than zero indicates the machine coding interprets the use of that word as evidence for *exclusion* of the response from the code. The larger the magnitude of a coefficient, the more impact the corresponding word will have on the trained model. We then evaluate the extent to which these coefficients align with the coding scheme. A lack of alignment implies that human and machine coders use fundamentally different linguistic coding rules. Next, we provide an example of comparing the highest and lowest coefficients to the inclusion and exclusion criteria in a human-generated coding scheme.

B. Evidence and examples

We examine the features (individual words, x_i) of our two trained models (one for the CR code and one for the L code) along with their associated coefficients, κ_i. In Tables I and II we show the words with the highest and lowest coefficients for the trained model for each coding scheme, which we compare to the inclusion and exclusion criteria in the associated coding schemes (described in Sec. II).

TABLE I. Coefficients for the trained model for the CR code. Left side (coefficient > 0) corresponds to inclusion criteria in the trained model and right side (coefficient < 0) corresponds to exclusion criteria. Words in bold are listed in the inclusion criteria for the human-generated coding scheme for this code (or share a root and are part of the same word family as a word in the inclusion criteria, e.g., “repetition” and “repeating”).

Word	Coefficient	Word	Coefficient
trial	3.01	analysis	−1.09
repeatable	2.23	accurate	−0.92
repeated	1.90	how	−0.90
multiple	1.80	model	−0.72
replicable	1.73	bias	−0.72
same	1.51	uncertainty	−0.71
lot	1.47	unbiased	−0.70
similar	1.32	statistical	−0.68
time	1.30	theoretical	−0.66
repetition	1.23	reviewed	−0.62
consistent	1.09	look	−0.61
replicability	1.08	possible	−0.56
reproducible	1.06	margin	−0.55
many	1.04	creating	−0.54
others	1.02	fair	−0.54
replicated	1.00	need	−0.53
repeatability	0.99	into	−0.52
reproducibility	0.83	properly	−0.52
result	0.82	wa	−0.51
whether	0.81	t	−0.49

For the CR code (Table I), the words with the highest coefficients are all either represented in the inclusion criteria or are close synonyms of words in the inclusion criteria of the human-generated coding scheme (for example, a “lot” is close to “multiple,” “similar” is close to “consistent”). Three of the twelve coding scheme inclusion criteria (1, 7, and 8) are not represented in the list of high coefficients. These three inclusion criteria are less common in the data and may be missed by machine coding. Looking at the distribution of coefficients, the trained model relies more on positive coefficient words compared to negative coefficient words (the magnitude of the positive coefficients are much larger than the magnitude of the negative coefficients), which supports the way human coders relied more on inclusion, rather than exclusion, criteria when coding for CR.

For the L code (Table II), the magnitudes of the negative coefficient words are much larger than we saw for the CR code. All exclusion criteria from the human-generated coding scheme are also represented by the words with low coefficients. This result again matches the human coding scheme, which relies almost exclusively on exclusion criteria, rather than inclusion criteria. Based on the list of words, two of the exclusion criteria (“vaguely worded

TABLE II. Coefficients for the trained model for the *L* code. Left side (coefficient > 0) corresponds to inclusion criteria and right side (coefficient < 0) corresponds to exclusion criteria. Words in bold are listed in the exclusion criteria for the human-generated coding scheme for this code (or share a root and are part of the same word family as a word in the exclusion criteria, e.g. “random” and “randomness”).

Word	Coefficient	Word	Coefficient
human	2.49	**distribution**	−1.41
friction	2.07	**randomness**	−1.15
measuring	1.56	**gravity**	−1.08
not	1.51	**distance**	−1.05
ruler	1.49	variation	−1.01
ball	1.38	motion	−0.97
different	1.25	statistical	−0.96
table	1.22	off	−0.95
source	1.10	density	−0.90
ramp	1.06	**quantum**	−0.90
resistance	1.05	would	−0.89
in	1.05	because	−0.89
material	1.03	**value**	−0.84
condition	1.03	count	−0.81
between	0.95	**random**	−0.79
difference	0.94	data	−0.79
incorrect	0.93	probability	−0.75
slightly	0.92	answer	−0.75
wind	0.89	**experimental**	−0.75
inconsistency	0.86	each	−0.74
initial	0.83	**average**	−0.74
too	0.82	**principle**	−0.73

answers" and "physical mechanisms that are not varying between experimental trials") are not obviously captured by the algorithm. These criteria, however, by definition do not have defining words, and all the examples explicitly listed in these exclusion criteria (e.g., gravity) are present in the low coefficient words.

C. Recommendations

We identify three key recommendations for evaluating a trained algorithm:

1. Evaluate the correspondence between the inclusion criteria of the coding scheme and the high (positive) coefficient words. Do all the words with a high coefficient correspond to an inclusion criteria? Are all inclusion criteria represented in the high coefficient list?
2. Evaluate the correspondence between the exclusion criteria of the coding scheme and the low (negative) coefficient words. Do words with low coefficients correspond to exclusion criteria? Are all exclusion criteria represented in the low coefficient list?
3. Compare the relative distribution of positive coefficients to the distribution of negative coefficients. Are the relative magnitudes appropriate based on the importance of inclusion versus exclusion criteria in the coding scheme and the answers to the previous questions?

If the answer to any of the questions is no, an element of the coding scheme may not be captured in the machine coding. In some cases, such as coding schemes with inclusion or exclusion criteria that are rare in the data, this misalignment may be acceptable as long as the researcher is aware of the omission when interpreting results from machine coding. If the researcher thinks this misalignment is unacceptable, consider expanding the size of the training set to include more examples of underrepresented criteria.

IV. STATISTICAL UNCERTAINTY

The second step in establishing the trustworthiness of an algorithm is to understand the statistical variability associated with applying the algorithm to a finite number of samples drawn from a larger population. Whenever we calculate the frequency of a code in a finite dataset, we are sampling only a subset of the total population. If we were to take repeated measurements of the frequency of a code in multiple samples drawn from the broader student population, we would find natural variation across the measurements. This variation occurs because each of the finite sample of responses features a degree of random variation. The random variation in these samples leads to variation in the frequency a single coder (whether human or machine) would measure between samples. The natural variation caused by these factors can be quantified by calculating the standard deviation of many randomly selected samples, applying principles of statistics to machine coding student data.

A. Explanation of method

To quantify the variation in the measurements made by machine coding, we calculate σ_{E_C}, the standard deviation of a distribution of values of E_C drawn from many different sample test sets of size n pulled from a larger test bank of size N_{bank}. We expect some fraction of the measurements of E_C to fall within one standard deviation of the mean $\overline{E_C}$ (approximately 68% if the distribution is normally distributed). We, therefore, define the uncertainty in a single measurement of E_C as the standard deviation of this distribution, as it captures the typical variability between individual measurements. While we can produce this estimate from repeated tests of a large dataset, we need a way for researchers to estimate the value of σ_{E_C} from a single dataset (such as when applying a previously trained algorithm to new data without human coding).

Fortunately, this estimation is possible because the standard deviation, σ_{E_C}, is characteristic of the phenomenon we are measuring. Because we are using an OVA approach, such that each response either contains (1) or does not contain (0) the code, our data are dichotomous. Thus, E_C, the computer's estimate of the frequency the code appears in a particular sample test set, can be represented as the mean of a Bernoulli distribution, where the number of times the code is present in the sample is $n_p = nE_C$ and the number of times the code is absent in the sample is $n_a = n(1 - E_C)$. The standard deviation σ_B of the Bernoulli distribution can be calculated as

$$\sigma_B^2 = \frac{1}{n}[n_a(0 - E_C)^2 + n_p(1 - E_C)^2] = E_C(1 - E_C). \quad (2)$$

The central limit theorem implies that E_C comes from a distribution with mean $\overline{E_C}$ and standard deviation $\bar{\sigma}_B/\sqrt{n}$, where $\bar{\sigma}_B$ is the average value of σ_B that emerges across all sample test sets. Thus, the statistical uncertainty in a single measurement of E_C is equal to $\bar{\sigma}_B/\sqrt{n}$. When only a single measurement of E_C has been taken (such as with new data not coded by humans) and $\overline{E_C}$ is unknown, we can approximate that $\overline{E_C} \approx E_C$ and estimate σ_{E_C} directly:

$$\sigma_{E_C} = \bar{\sigma}_B/\sqrt{n} = \sqrt{\frac{\overline{E_C}(1 - \overline{E_C})}{n}} \approx \sqrt{\frac{E_C(1 - E_C)}{n}}. \quad (3)$$

This estimate can be computed with a single sample test set, as it requires only the measurement E_C in a single sample test set and the number of responses n in that sample test set. Importantly, this statistical uncertainty can be applied to a machine-coded sample test set *without any new human coding* beyond that used to initially train the model. Below, we evaluate this estimate of statistical

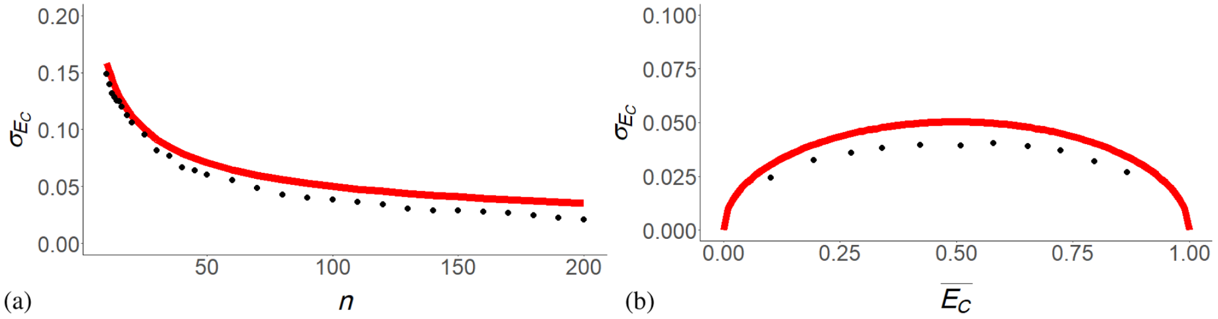

FIG. 4. Comparison between empirical data (black points) and Eq. (3) (red lines). (a) Standard deviation of E_C as n, the number of responses in each sample test set, varies empirically (code: CR, $p_{\text{train}} = 0.5$, $\overline{E_C} \approx 0.5$, and the proportion of responses in the test bank that contain the code is 0.5). (b) Standard deviation of E_C as $\overline{E_C}$ varies empirically (code: CR, $n = 100$, $p_{\text{train}} = 0.5$).

uncertainty with empirical results to demonstrate its robustness to finite sampling.

B. Evidence and examples

We assess the validity of Eq. (3) in two steps. First, we evaluate the relationship between σ_{E_C} and n. Second, we evaluate the relationship between σ_{E_C} and $\overline{E_C}$.

To evaluate the relationship between σ_{E_C} and n (sample test set size), we train an algorithm for the CR code using a training set of size 600 responses with $p_{\text{train}} = 0.5$ (as per the recommendation to balance the training set, described above in the methods section). We test 30 different values of n between 10 and 200. For each value of n, we draw 1000 sample test sets of size n from a test bank where the proportion of responses containing the code is also 0.5 (so that we are controlling for variability due to proportion of responses in the test set sample). We then have our algorithm compute E_C for every sample. For each value of n, we calculate an empirical value for σ_{E_C} by computing the standard deviation across the 1000 samples.

We find that the empirical values of σ_{E_C} approximately follow the predicted $1/\sqrt{n}$ relationship [see Fig. 4(a)], though the empirical values are consistently less than the values predicted by Eq. (3). This overestimation may be because our test set samples are pulled from a test bank of finite size ($N_{\text{bank}} = 300$), so the variability does not quite reach the level we would expect if our samples were pulled from a large population. That is, the number of ways data may vary in different samples is diminished to some extent by the fact that the test bank is only a few times larger than n. Fortunately, Eq. (3) overpredicts, rather than underpredicts, the standard deviation, meaning the calculated σ_{E_C} from a single test set sample will be a conservative estimate of statistical variability.

To evaluate the relationship between σ_{E_C} and $\overline{E_C}$, we test values of $\overline{E_C}$ between 0 and 1. We use the same algorithm for the CR code with a training set of size 600 and $p_{\text{train}} = 0.5$. We create a set of test banks where the proportion of responses containing the code (as determined by the human coder) ranges from 0 to 1 inclusive in steps of 0.1. We draw repeated samples from each test bank; each of these values corresponds to a fixed value of $\overline{E_C}$ (these values range from about 0.1 to 0.9 as systematic effects narrow the range). For each value of $\overline{E_C}$, we take 1000 sample test sets of size $n = 100$ and calculate σ_{E_C} across the 1000 samples.

We find that the empirical values of σ_{E_C} again approximately follow the predicted parabolic relationship [see Fig. 4(b)]. As seen in the relationship to n, the empirical values are consistently less than the values predicted by Eq. (3), likely for the same reason. Again, the analysis shows that Eq. (3) provides a conservative estimate of statistical variability.

In Appendix B, we demonstrate the validity of estimating σ_{E_C} through a measurement of E_C from a single sample test set rather than the mean $\overline{E_C}$. We also show, for multiple coding schemes and values of n, that this statistical variability is independent of the size of the training set (Fig. 12). This independence of variability with training set size reflects that the statistical variability comes only from sampling a subset of responses from the larger test set. This means that σ_{E_C} is constant even if the size of the training set is so small that past work would suggest the machine coding would be quite poor [17].

Finally, we provide an example of calculating statistical uncertainty using this method. Consider a dataset of 160 responses that have not been coded by humans. Without performing any human coding, we can apply our trained model to the 160 responses. Say the trained model applies the code to 96 of the 160 responses. This means that $E_C = 96/160 = 0.6$. We take this value of E_C as an approximation of $\overline{E_C}$. Therefore, $\sigma_{E_C} \approx \sqrt{0.6 \times 0.4/160} = 0.039$.

In summary, σ_{E_C} can be conservatively estimated from two characteristics of a sample test set: the computer's estimate of the presence of the code in the sample test set, E_C, and the number of responses in the sample test set, n. The researcher can then report their estimate of the presence of the code as $E_C \pm \sigma_{E_C}$.

C. Recommendations

The analysis above leads to a single recommendation for estimating statistical uncertainty:

1. Whenever reporting the frequency of a code in a sample estimated by machine coding, report the statistical uncertainty using Eq. (3).

We note that it is possible to do this without any human coding beyond the initially human-coded training set.

V. SYSTEMATIC UNCERTAINTY FROM THE TRAINED ALGORITHM

The third step in establishing the trustworthiness of an algorithm is to understand any potential systematic effects in how the algorithm codes data. When comparing two coders, such as a machine coder and a human coder or even two human coders, the two coders can apply the same coding scheme in consistently different ways, leading to systematic (as opposed to random) uncertainty. In this section, we demonstrate how systematic effects, wherein a machine coder can systematically over- or underestimate the presence or absence of a code, emerge due to an outcome imbalance in the sample test set (E_C). We then show how this systematic effect can be measured and thus used to calibrate a measurement from a trained algorithm. For now, we consider only systematic effects when applying an algorithm to data from a similar population. The next section will explore impacts due to data from different populations.

A. Explanation of method

We can evaluate potential systematic offsets between the human and machine coder by calculating the difference between an individual measurement of E_C and the human estimate E_H. We observe that the computer systematically over- or underestimates the presence of a code depending on the prevalence of the code in the test bank. We can estimate this relationship between E_C and the systematic uncertainty $E_C - E_H$ through a predictive linear model, which we define as a function $S(E_C)$. Following the determination of this best fit function $S(E_C)$ from a set of human-coded test banks, researchers can then apply this function to measurements without further human coding. We assume that the human coder is "correct" whenever the human and machine coders disagree, such that we calibrate the machine coder to match the human coder (rather than the other way around).

B. Evidence and examples

We calculate the systematic effect with trained models for both the CR and L codes. Following the recommendation above, we fix $p_{\text{train}} = 0.5$, which has the additional benefit of controlling for a relationship between p_{train} and the systematic effect (as observed in Sec. II E). We generate a set of eleven different test banks where the proportion of responses containing the code varies from 0 to 1 inclusive in steps of 0.1. The size of each test bank is $N_{\text{bank}} = 200$. For each test bank, we compute E_C and E_H for 100 sample test sets of size $n = 100$ pulled from the test bank.

We observe a systematic difference between the human and machine coder that tends to be more extreme with more extreme values of E_C (Fig. 5). For the CR code [Fig. 5(a)], the systematic effect is such that the model overestimates the prevalence of the code when $E_C < 0.5$ and underestimates the prevalence of the code when $E_C > 0.5$. For the L code [Fig. 5(b)], the direction of the systematic effect is the same, but the boundary between over- and underestimation is shifted: the model overestimates the prevalence of the code when $E_C < 0.3$ and underestimates the prevalence of the code when $E_C > 0.3$. Variability around the average systematic uncertainty falls within the previously defined statistical uncertainty, as 95% of the data falls

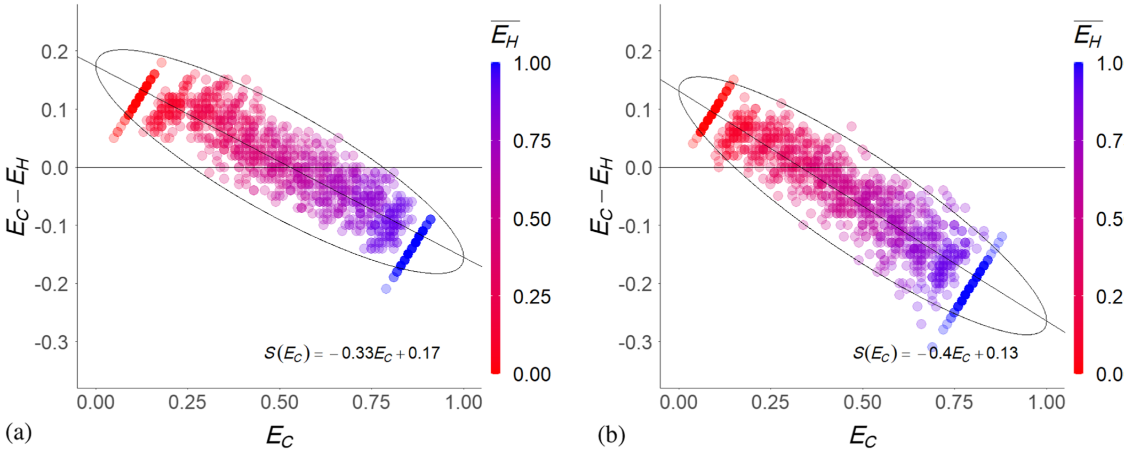

FIG. 5. Given a value of E_C, we compute the systematic uncertainty using the line of best fit $S(E_C)$. Scatter around the line of best fit is within the statistical uncertainty, with at least 95% of the data falling within the black oval depicting $2 \times \sigma_{E_C}$ [see Eq. (3)]. (a) $S(E_C)$ for the CR code (b) $S(E_C)$ for the L code.

within two standard deviations of σ_{E_C} as calculated by Eq. (3) (indicated by the black ovals in Fig. 5). These results imply that a measurement of E_C is subject to a systematic error that itself is dependent on the value of E_C.

The effect is also dependent on the trained model, which we demonstrate in Appendix C (Fig. 13). There, we see that when we construct new trained models for the CR and L codes using a different (yet still balanced) training set, the best fit lines, $S(E_C)$, have the same overall downward linear shape but have different parameters than the fit equations in Fig. 5. The estimates for the systematic, therefore, depends on the training set used (i.e., is unique to the trained model). The analysis also shows that the difference in systematic uncertainty for two different trained models can go beyond the statistical uncertainty, especially at extremes. We expect that this occurs because each trained model learns slightly different coding rules.

To account for this source of systematic uncertainty in subsequent analysis, a researcher can use the best fit function for their trained model to find the systematic offset that corresponds to their measured value of E_C for a new dataset (that has not been coded by humans). One should report the estimate of $S(E_C)$ for their estimate of E_C along with its corresponding statistical uncertainty as $E_C \pm \sigma_{E_C} - S(E_C)$. For example, for a measurement of $E_C = 0.75$ with the CR code [and the trained algorithm that produced Fig. 5(a)], we would use the equation for $S(E_C)$ in Fig. 5 to obtain $S(E_C = 0.75) = -0.33 \times 0.75 + 0.17 = -0.08$.

C. Recommendations

We identify two key recommendations for determining the systematic effects in machine coding:

1. For a single trained model applied to new data from a similar population, measure the best fit equation $S(E_C)$ for the systematic uncertainty as a function of the machine coding estimate, E_C, using a test bank of human-coded data. The larger the test bank size and sample test set size and the larger the spread in values of E_C, the more confident you will be in the function.
2. Report the systematic uncertainty alongside your estimate and its statistical uncertainty in the form $E_C \pm \sigma_{E_C} - S(E_C)$.

If the magnitude of the systematic uncertainty of your result is large enough that it calls your results into question, improve your trained model with more data or through improvements to the natural language processing and machine learning.

We return to the question of how much hand-coded data are needed to measure systematic uncertainty in this way. In our analysis of the CR and L codes, we used test data that included at least 200 responses that contained the code and at least 200 responses that did not (separate from the 600 responses included in the training set). This full amount of 400 responses may or may not be necessary for researchers to evaluate systematic effects, depending on the particular code. One way this number could be reduced is by reducing the range of $\overline{E_H}$ values in the test banks, though there is a trade-off as this will reduce precision in the calculated systematic uncertainty value. In Fig. 5 we use the full available range of values, as we include a test bank with $\overline{E_H} = 0$ and a test bank with $\overline{E_H} = 1$. If the prevalence of a code ranges from, say 0.2 to 0.6, and depending on educational conditions, it may not be necessary to calculate the best fit function $S(E_C)$ on this full range. Alternatively, one can reduce the size of the test banks at each value of $\overline{E_H}$, which also reduces precision but would maintain the full range.

VI. SYSTEMATIC UNCERTAINTY IN NEW DATASETS

The fourth step in establishing the trustworthiness of an algorithm is to evaluate any potential systematic effects based on how an algorithm may code data that comes from a distinct population of, in our case, student responses than those used in the initial training. Quantitatively estimating this source of potential systematic offsets is particularly important for physics education researchers looking to use machine learning and natural language processing at scale, such that a model trained on data collected at a particular time, institution, course, or course level could then be applied to large quantities of data from other contexts with limited additional human coding. In this section, we demonstrate how to estimate such systematic effects using similar methods to those in the previous section.

A. Explanation of method

In the previous section, we demonstrated that systematic effects between the human and machine coder can be written as a function of E_C when the training set and test set are pulled from the same population. Additional systematic effects may arise, however, when the characteristics of students in the test set are different from those represented in the training set [50]. These characteristics can impact the machine coding if, for example, the new population of students use language differently than the initial dataset. For example, students may use unique jargon or special terms that a particular instructor introduced in a particular physics class, but not in others. There are, of course, many other ways in which the responses by one population of students may differ systematically from those of another population.

To account for this additional systematic effect, we determine the function $S(E_C)$ using human-coded test data from the new population. When reporting results, accounting for the systematic effects then follows the steps from the previous section. Though some new human coding is needed to complete this step, we emphasize that this method limits human coding to only the amount needed

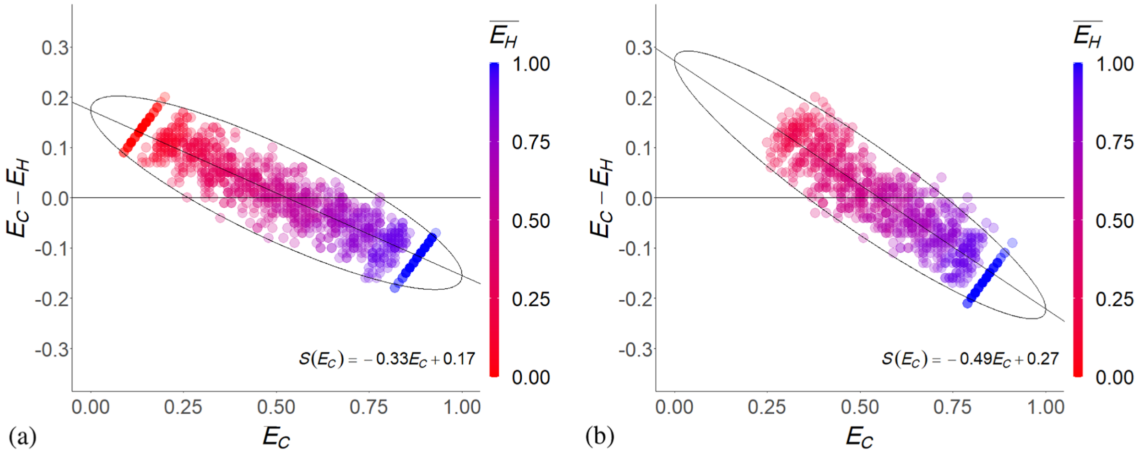

FIG. 6. Sample test sets from the same population as the training set have lower systematic uncertainties than sample test sets from a different population as the training set. The training set is identical in both plots, consisting of a 20/80 mix of pre-post responses. (a) $S(E_C)$ with sample test sets from same population as the training set (test bank is 20/80 mix of pre or post responses). (b) $S(E_C)$ with sample test sets from different population as the training set (test bank is all preresponses). (Code = CR, $p_{\mathrm{train}} = 0.5$).

to establish trust in the researchers' ability to measure systematic effects in the new dataset. Next, we will provide an example of estimating the function $S(E_C)$ with data from a new population and we will compare it to an estimate of $S(E_C)$ with data from the same population as the training set.

B. Evidence and examples

Our *trustworthy* dataset includes several semesters where the *trustworthy* survey question was asked only at postsurvey. Because the survey question is related to course material for the populations being surveyed, we expect that there are meaningfully different characteristics between the presurvey and postsurvey responses. This provides an opportunity to test for population systematic effects by assuming that responses from students at presurvey represent a distinct population than responses from students at postsurvey. To do so, we generate a trained model for the CR code with a training set size of 500 majority postsurvey responses (in this instance we had to reduce training set size and could not use exclusively postsurvey responses because of limited data). The training set is comprised of 80% postsurvey responses and 20% presurvey responses. In accordance with the recommendations above, we balance the training set such that half the responses contained the CR code according to human coders.

Using nontraining data responses remaining in the dataset, we create two sets of test data: the first set (set Pre-Post) mirrors the structure of the training set with 80% postsurvey and 20% presurvey responses, while the second set (set Pre) includes only pre-survey responses. Thus, set Pre-Post is considered the same population as the training set and set Pre is considered a different population.

We calculate the fit $S(E_C)$ as a function of the computer's estimate E_C using the methods outlined in the previous section. We construct test banks where the proportion of responses that contain the code varies from 0 to 1. Each test bank is of size $N_{\mathrm{bank}} = 200$. From each test bank, we take sample test sets of size $n = 100$. We could not generate samples with values of E_C lower than about 0.3 using set Pre because of insufficient presurvey data.

As per the previous section, we again find a linear trend between the systematic uncertainty $E_C - E_H$ and the computer's estimate E_C. We find the systematic uncertainties for set Pre-Post are given by the fit line $S(E_C) = -0.329 \times E_C + 0.175$ [Fig. 6(a)] and the systematic uncertainties for set Pre are given by the fit line $S(E_C) = -0.494 \times E_C + 0.273$ [Fig. 6(b)]. Across the range of possible values of E_C, the systematic uncertainty tends to be farther from zero when machine coding samples from the set Pre, a different population than the training set, compared to when machine coding samples from the set Pre-Post, the same population as the training set. We also see that the statistical variability is the same for both set Pre and set Pre-Post and that 95% of the data fall within two standard deviations as calculated by Eq. (3) (given by the black ovals in Fig. 6).

We have presented just one example of the systematic differences that may arise when the characteristics of students in the test set are different from those represented in the training set. The results motivate caution to be taken in all instances involving changes to population characteristics between training and test data, including changes in the composition of, for example, student major, institution, gender, race and ethnicity, or international student status. Many variables can affect the way language is used and thus how machine (and human) coders interpret it. We have shown, however, that systematic differences can be measured by hand coding a relatively small set of data from the new population. Future work should continue to investigate the effects that arise from applying machine coding to different populations of students.

C. Recommendations

We identify one key recommendation for determining the systematic effects in applying a machine coding algorithm to a new population:

1. When applying a trained model to a new student population, regenerate $S(E_C)$ by hand coding a subset of the test data from the new population. Then follow the recommendations of the previous section using this new function $S(E_C)$.

Additional hand-coded data are needed to measure systematic effects in the new dataset. The amount of data needed will be comparable to the amount of data needed to measure systematic uncertainty from the trained algorithm in Sec. V, though it may be possible to use less data if it can be shown with a smaller amount of data that systematic effects are not greater than the previously calculated systematic uncertainty in the trained algorithm. In this case, the previously calculated function $S(E_C)$ may be used to compute the systematic.

VII. FULLY WORKED EXAMPLE

As a demonstration of this methodology altogether, this section presents a fully worked example with a different code than in the previous analyses. In what follows, we train a new model, evaluate the optimized model, and calculate the statistical and systematic uncertainties associated with an estimate of the code in a new test set without additional human coding, following the methodology summarized in Fig. 1.

For this example, we use a different code from the coding scheme for the *trustworthy* data: Uncertainty, abbreviated U. Two human coders achieved a Cohen's kappa value of 0.9 in 10% of the data for this code. Then one coder coded the rest of the data. The U code is defined as "measures were taken during the procedure to reduce or account for error or uncertainty and/or there is a small calculated uncertainty." The inclusion criteria for the U code are

1. uncertainty in measurements,
2. uncertainty of the results,
3. methods serve to reduce error,
4. error bars or bounds,
5. sources of error in measurements,
6. signal-to-noise ratio,
7. accounting for or reducing systematic error, random error, or "human error" in the process of taking measurements.

We use four distinct data groupings for this analysis: a training set (400 pre- and postsurvey responses from semesters prior to Fall 2022), a new dataset not coded by humans (286 postsurvey responses from Fall 2022), and two sets of test data. The first set of test data (labeled set Pre-Post) includes responses from a similar population to those in the training set data but different from those in the new (un-coded) data (270 pre and postsurvey responses from semesters prior to Fall 2022). The second set of test data (labeled set Post) includes responses from a different population to those in the training data but similar to those in the new (uncoded) data (270 postsurvey responses from semesters prior to Fall 2022). The responses for the training data and the test data were all hand coded but the new (postsurvey from Fall 2022) dataset was not.

We first trained an algorithm for the U code using a training set of size 400. We set $p_{\mathrm{train}} = 0.5$ as per the recommendations (in this instance we had to reduce training set size because of limited data, as instances of the U code were less common).

A. Evaluate the trained model

Table III presents the coefficients associated with the trained model. The highest weighted words are uncertainty and error, which are part of six of the seven inclusion criteria for this code, such as accounting for or reducing error, human error, and sources of error. One inclusion criterion, "signal to noise ratio," occurs rarely in the dataset and does not appear in the top weighted words. Though the U code does not have explicit exclusion criteria, percent error was part of the inclusion criteria for a different code and was understood by the human coders to not be related to the inclusion criteria for the U code. Accordingly, percent is the one of the words with the lowest (most negative) coefficients. Overall, the trained model seems to accurately capture our coding scheme based on these coefficients.

B. Calculate statistical uncertainty

We apply the trained model to $n = 286$ responses in the uncoded dataset. This machine coding returns a measurement of $E_C = 0.409$. With this measurement,

TABLE III. Coefficients for the trained model for the U code. Left side (coefficient > 0) corresponds to inclusion criteria and right side (coefficient < 0) corresponds to exclusion criteria. Words in bold are listed in the inclusion criteria for the human-generated coding scheme for this code (or share a root and are part of the same word family as a word in the inclusion criteria, e.g., uncertainty and the common typo "uncertainities").

Word	Coefficient	Word	Coefficient
uncertainty	4.76	percent	−0.97
error	3.09	this	−0.58
low	1.07	within	−0.56
human	0.92	fit	−0.50
source	0.84	it	−0.49
uncertainities	0.70	likely	−0.47
account	0.68	consistent	−0.47
minimized	0.60	theoretical	−0.46
possible	0.58	analysis	−0.44
accounted	0.55	peer	−0.42
reduce	0.53	same	−0.39

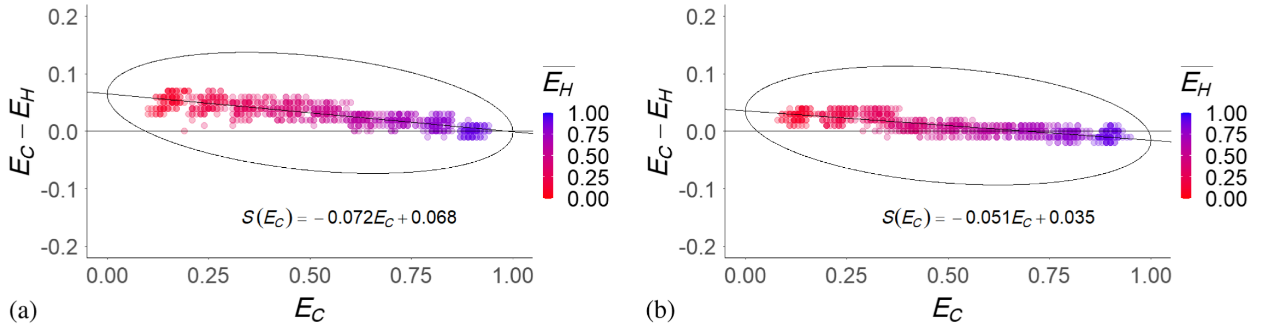

FIG. 7. Trained model for the U code applied to sample test sets from two different sets of test banks: (a) $S(E_C)$ with set Post-Pre, and (b) $S(E_C)$ with set Post (Code = U, $p_{\mathrm{train}} = 0.5$. Black oval depicts $2 \times \sigma_{E_C}$).

we calculate the statistical uncertainty from Eq. (3), giving $\sigma_{E_C} = \sqrt{0.409 \times (1 - 0.409)/286} = 0.029$.

C. Calculate systematic uncertainties

To calculate systematic uncertainty associated with the new uncoded data, we first need to determine a systematic uncertainty function from the hand-coded test data. First, we divide set Pre-Post (where data come from a population similar to the training data but different from the uncoded data) into nine different test banks, each with a different proportion of responses containing the code (according to human coders) between 0.1 and 0.9 inclusive. The size of each test bank was $N_{\mathrm{bank}} = 150$. For each test bank, we compute E_C and E_H for 100 sample test sets of size $n = 100$ pulled from the bank. Again, because U is a less common code, we could not make larger banks without losing the ability to estimate $S(E_C)$ on as wide a range between $E_C = 0$ and 1 as possible.

We generate a plot of $E_C - E_H$ versus E_C [Fig. 7(a)] for the set Pre-Post data. The systematic uncertainties are measured with the fit line $S(E_C) = -0.072 \times E_C + 0.068$ [Fig. 7(a)]. The systematic uncertainties follow the same general pattern (negative slope) seen previously, but the range is smaller than for codes we discussed previously. The observed statistical variability of $E_C - E_H$ is also much smaller than the statistical uncertainty of E_C, with nearly all data falling well within 2 standard deviations (given by the black oval). We expect that this reduced variability in $E_C - E_H$ reflects that the code is particularly and confidently captured by the trained model.[1]

We then perform the same analysis using set Post, where data come from a population different from the training data but similar to the uncoded data. In this case, the systematic uncertainty can be measured with the fit line $S(E_C) = -0.051 \times E_C + 0.0353$ [Fig. 7(b)]. We again see the same linear trend, but now with a slightly smaller slope magnitude and a smaller intercept. This analysis demonstrates that, in this example, perhaps counterintuitively, the systematic effect is smaller when estimated using hand-coded test data from a population *different* from the training data, though this difference is indistinguishable given the statistical uncertainty.

After doing this preparatory work, we calculate the systematic uncertainty of the trained model with our estimate of $E_C = 0.409$. The systematic uncertainty from the set Pre-Post analysis is $S(E_C = 0.409) = -0.072 \times 0.409 + 0.068 = 0.039$. Next, we calculate the systematic uncertainty using set Post because our uncoded data are postsurvey data. The systematic uncertainty from the set Post analysis is $S(E_C = 0.409) = -0.051 \times 0.409 + 0.0353 = 0.014$.

D. Reporting the result

Based on our methodology, we would report that the frequency of the U code in the new dataset is $E_C = 0.409 - 0.014 \pm 0.029$. We opt to use the estimate of systematic uncertainty from set Post rather than set Pre-Post because the student population in set Post is more similar to the population in the new dataset. The difference between the two calculations of systematic uncertainty are indistinguishable within the larger statistical uncertainty, so we consider the estimate of systematic uncertainty from set Post to also capture any systematic uncertainty from the trained algorithm.

The skeptical reader may ask if this measurement aligns with one made through human coding of the same dataset. We do not wish to promote comparison with human coding as a validation method because it eliminates our ability to use machine coding as a tool to improve efficiency. We can report, however, that upon skimming through the spreadsheet of the machine's codes, the machine coding

[1] The ovals in Fig. 7 represent the statistical uncertainty of E_C, while the variability in the data about the line of best fit reflects the statistical uncertainty of the quantity $E_C - E_H$. The variability of E_C—E_H should be less than or equal to our prediction for the statistical uncertainty of E_C because E_C and E_H are not independent variables (they are correlated). The purpose of adding the ovals is not to provide a theoretical prediction for the variability in $E_C - E_H$, but rather to provide a check that the variability in the systematic uncertainty of the measurement E_C does not exceed what is already being reported as the statistical uncertainty for the measurement E_C.

of individual responses appeared correct. Nonetheless, to appease the skeptical reader, we human coded the first 60 responses in the uncoded new dataset and measured $E_H = 0.419 \pm 0.064$, where we calculate statistical uncertainty using Eq. (3). This result is well within uncertainties (both statistical and systematic) of the computer estimate.

It is notable that the systematic uncertainty for the U code (Fig. 7) is significantly reduced compared to the systematic uncertainty for the codes we examined earlier (Fig. 5). Furthermore, the level of actual statistical variation about the systematic uncertainty fit is smaller than in the systematic uncertainty fits for the other codes (Fig. 5). We interpret this reduced uncertainty to be a property of the U code, specifically. That is, the trained model was able to better capture the U code, likely because sentences that contain this code tend to be more similar to one another compared to the CR and L codes.

VIII. FUTURE WORK

We have presented a framework for measuring and reporting quantitative education claims with machine coding of student text data. This framework was developed using two example datasets, the *trustworthy* dataset and the *sources* dataset, and only a handful of individual codes. The limitations of human coding constrained the number of datasets and codes we could include in this analysis. As this framework continues to be applied, new data sources and coding schemes should be used to reveal nuances that improve and build upon our recommendations.

There are specific use cases of supervised natural language processing where these methods cannot be used. Most notably, these methods cannot be used to assist in making inferences about individual students, because these methods are designed to estimate the prevalence of a code on a population level and to compute statistical and systematic uncertainty of this estimate. Rather, these methods should be used to evaluate samples of responses from many students, such as individual classes.

The analyses above also relied on a one-vs-all approach, where each code was evaluated independently and a response could be coded for any number of codes. It is possible to use a one-vs-all approach with a coding scheme that uses multiple mutually exclusive codes, as in Ref. [15]. There may be some cases, however, where it is necessary to enforce the constraint that the prevalence of these mutually exclusive codes must add to one. Future work should evaluate and adapt our methods for such cases. The estimate of statistical uncertainty, for example, would be different because the statistical processes would come from a different probability distribution than the Bernoulli distribution.

Our machine coding algorithms also used logistic regression and our method for evaluating the correspondence between the trained model and human-generated coding scheme in particular is based in the logistic regression algorithm. Future work may use and expand upon these methods by using other algorithms, such as RandomForest, SVM, and neural networks. Additionally, future work should investigate empirically if statistical and systematic uncertainty are impacted by the use of these other algorithms. One particularly promising avenue, as pretrained large language models (LLMs) become available, is to integrate LLMs into the construction of trained models that classify PER data. Evaluating the correspondence between trained models that use LLMs and a human-generated coding scheme may pose a challenge because you cannot read off parameters associated with unique features as you can for logistic regression. This may be worthwhile, however, as the LLM approach could reduce the amount of training data needed while also reducing systematic uncertainty. With the LLM approach, it is still necessary to account for statistical and systematic uncertainty in analysis and we expect the methods above would apply.

A key constraint on the methods throughout is the amount of hand-coded data required. An untested idea is that generative AI could be used to generate additional responses that do or do not contain a code (in a process similar to efficient mass creation of isomorphic physics problems [55]). These additional responses could particularly be used to create much larger balanced training sets and larger test sets for assessing systematic effects. The methodological and ethical considerations that would go into this process are beyond the scope of this paper, but we would be interested in seeing this idea explored in future work.

We focused our analysis of systematic effects from the trained algorithm on the role of outcome imbalance in the test set. Other sources of systematic uncertainty that we did not explore may also exist, which is another area of future study. In addition, we demonstrated how a difference in student characteristics (whether they are taking a presurvey or a postsurvey) impacts the level of systematic effects. Future work should investigate explicitly the extent to which other characteristics (gender, race, institution, course style) measurably impact systematic uncertainty for a range of data sources. For example, AI detectors have been shown to systematically label writing from non-native English speakers incorrectly as AI generated [56], so similar systematic errors are likely to occur when using machine coding in PER.

Last, there may be ways to reduce the amount of additional hand-coded data needed to compute systematic error while making trustworthy claims. For example, a researcher may be able to dramatically reduce the amount of hand-coded data needed for evaluating systematic effects if they are willing to accept the cost of increased systematic error attributed to their claims. A researcher may choose not to find the best fit function $S(E_C)$ for a particular code and student population group if they can instead provide evidence to suggest that the systematic uncertainty is well within the statistical variability from random sampling. For example, it could be argued using approximately 50 responses that the U code has a systematic uncertainty much less than the statistical uncertainty. As with any research that seeks to make claims about large groups of students, there is a fundamental tension between the additional effort needed to gather additional data and the fact that gathering additional data will usually improve the trustworthiness, precision, and accuracy of claims made with the data. By using these methods based in uncertainty quantification, researchers may stop collecting data and/or human coding as soon as sufficient trustworthiness is reached.

IX. CONCLUSION

We have presented a four-part methodology that applies established scientific tools and procedures (evaluating models, calculating statistical uncertainty, calculating systematic uncertainty) to build trust in effective machine coding. We demonstrated this method is effective for more than one coding scheme and provided a real-world example of using our method to machine code data from a new dataset without additional human coding. We propose (and provide evidence for) several recommended best practices in machine coding: balancing training sets across code outcomes, drawing comparisons between machine coding mechanisms and human coding mechanisms (coding schemes), and measuring and accounting for statistical and systematic uncertainty whenever a quantitative claim is made. We hope these recommendations can continue to evolve as the use of machine learning and natural language processing is applied to PER data.

ACKNOWLEDGMENTS

We thank Matthew Dew, Lauren Douglass, Daniella Garcia Almeida, Sophia Jeon, and Ali Mazrui for assistance with hand-coding. Thank you to Nicholas T. Young for your data science expertise and detailed feedback, and thank you to Tor Ole Odden, Meagan Sundstrom, Marcos D. Caballero, Carl Wieman, and Megan Flynn for insightful comments on early versions of this article. This work was supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2139899 and National Science Foundation Grants No. DUE-1808945 and No. DUE-2000739.

APPENDIX A: EFFECT OF TRAINING SET IMBALANCE

1. Training set balance for the limitations code

In the main text, we presented evidence with the CR code in the *trustworthy* dataset that balanced training sets (i.e., 50% of the responses include the code) are most effective for training a model. We additionally assess this effect for the L code in the *sources* dataset with the same procedure. We train our algorithm with nine different training sets, each with a different value of p_{train} between 0.1 and 0.9 inclusive. The size of the training set for each algorithm was 600. For each algorithm, we compute E_C and E_H by pulling 100 sample test sets of size $n = 100$ from a test bank of size $N_{\text{bank}} = 200$. We fix the proportion of responses in the test bank containing the code to 0.5.

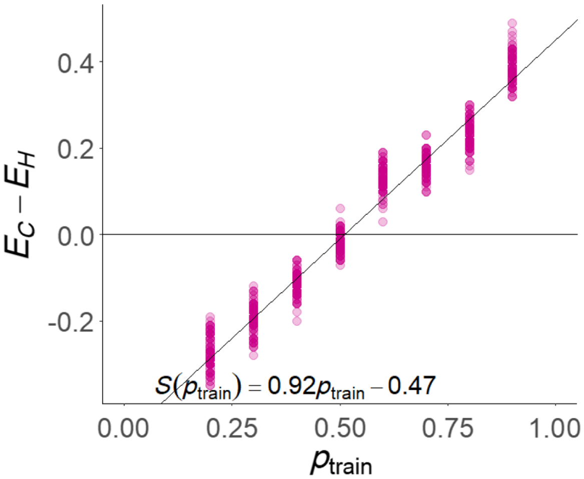

FIG. 8. Systematic uncertainty increases as p_{train} is less balanced ($p_{\text{train}} = 0.5$). (Code: Limitations, proportion of responses containing the code in the test bank is 0.5).

Figure 8 shows that the difference between E_C and E_H is zero for $p_{\text{train}} = 0.5$ but deviates from zero as p_{train} deviates from 0.5. Given that this finding is consistent for two different codes from distinct datasets, we suggest the finding is likely generalizable to other codes as well, though future work should continue to test this empirically.

2. Training set balance for varying proportions of responses in the test bank that contain the code

In the main text, we demonstrated the benefit of balancing a training set using a test bank where the proportion of responses in the test bank containing the code is also balanced (50% of the responses contain the code). In this section, we demonstrate what happens at other proportions of responses in the test bank.

We test values of p_{train} between 0.1 and 0.9 (in steps of 0.1) for two different test sets with different proportions of responses that contain the code: (a) 0.3 and (b) 0.7. We use the same algorithm for the CR code used above with a training set of size 600. For each value of p_{train}, we take 100 sample test sets of size $n = 100$ and estimate $S(E_C)$ across the 100 samples. These data are displayed in Fig. 9.

We find that for the test bank where the proportion of responses that contain the code is 0.3, $S(p_{\text{train}})$ is zero around $p_{\text{train}} = 0.4$. For the test bank where the proportion of responses that contain the code is 0.7, $S(p_{\text{train}})$ is zero around $p_{\text{train}} = 0.6$. Thus, the optimal value of p_{train} has some correlation with the proportion of responses in the test bank that contain the code.

In spite of the correlation between the optimal value of p_{train} and the proportion of responses in the test bank that contain the code, we argue that a balanced training set with $p_{\text{train}} = 0.5$ is the best choice for training a model. First, in

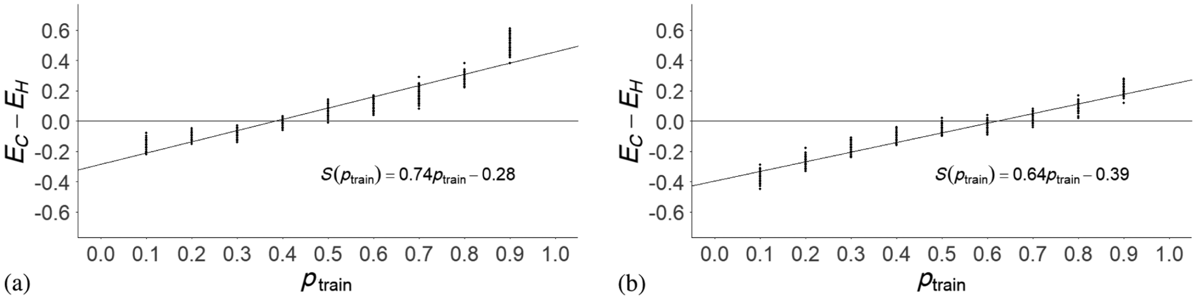

FIG. 9. (a) $S(p_{\text{train}})$ where the proportion of responses in the test bank that contain the code (as determined by human coders) is 0.3, (b) $S(p_{\text{train}})$ where the proportion of responses in the test bank that contain the code (as determined by human coders) is 0.7.

both of our test banks in Fig. 9, the systematic uncertainty is within statistical uncertainty to zero when $p_{\text{train}} = 0.5$. Second, when constructing a training set, we do not know the proportion of responses in the new datasets to which the model will be applied, and thus cannot make *a priori* decisions about the best proportion to use in the training set based on the correlation with the new dataset. Thus, we must determine what value of p_{train} is optimal *on average* across all possible proportions of responses that contain the code in the test set. Based on the analyses here, we recommend balancing the training set (fix $p_{\text{train}} = 0.5$) and modeling the remaining systematic uncertainty using $S(E_C)$ (with the process described in Sec. V).

APPENDIX B: STATISTICAL UNCERTAINTY

1. Validity of statistical uncertainty expression for the L code

In the main text, we present evidence with the CR code that Eq. (3) describes the statistical variability of the machine coded responses. Here we assess the applicability of Eq. (3) to the L code using a training set of size 600 with $p_{\text{train}} = 0.5$, using the same procedures as with the CR code in the main text.

We first probe the relationship between σ_{E_C} and n by testing 30 different values of n between 10 and 200. For each value of n, we draw 1000 test set samples of size n from a test bank where the proportion of responses containing the code is 0.5. We then have our algorithm compute E_C for every sample. For each value of n, we calculate an empirical value for σ_{E_C} by computing the standard deviation across the 1000 samples. As for the CR code, we find that the empirical values of σ_{E_C} approximately follow the predicted $1/\sqrt{n}$ relationship but that Eq. (3) slightly overestimates σ_{E_C} [Fig. 10(a)].

We then probe the relationship between σ_{E_C} and $\overline{E_C}$ for the L code. We test values of $\overline{E_C}$ between 0 and 1. We create a set of test banks where the proportion of responses containing the code ranges from 0 to 1 inclusive in steps of 0.1. We draw repeated samples from each test bank to calculate $\overline{E_C}$ for each test bank. For each value of $\overline{E_C}$, we take 1000 sample test sets of size $n = 100$ and calculate σ_{E_C} across the 1000 samples. As for the CR code, we find that the empirical values of σ_{E_C} approximately follow the predicted parabolic relationship but that Eq. (3) slightly overestimates σ_{E_C} [Fig. 10(b)].

2. Validity of the estimate $\sqrt{\frac{\overline{E_C}(1-\overline{E_C})}{n}} \approx \sqrt{\frac{E_C(1-E_C)}{n}}$

Here we assess the quality of the estimate $\sqrt{\frac{\overline{E_C}(1-\overline{E_C})}{n}} \approx \sqrt{\frac{E_C(1-E_C)}{n}}$ that is made in Eq. (3). We check the assumption as a function of n and as a function of $\overline{E_C}$.

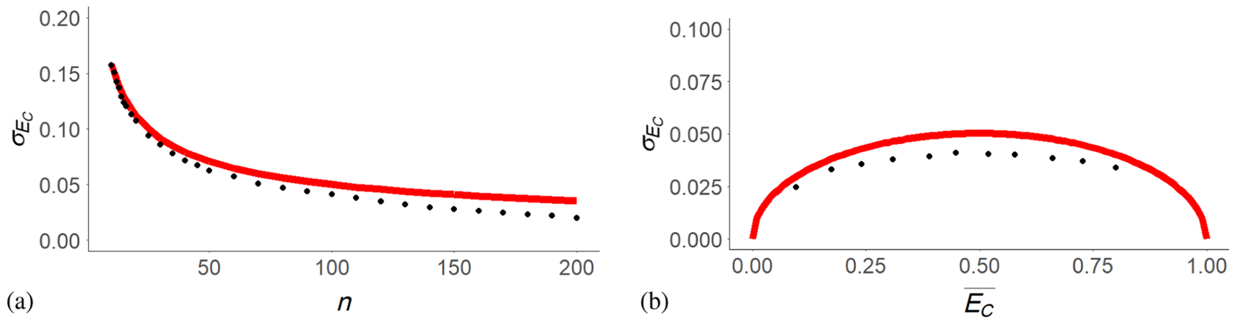

FIG. 10. Comparison between empirical data (black points) and Eq. (3) (red lines). (a) Standard deviation of E_C as n, the number of responses in each sample test set, varies empirically (code: L, $p_{\text{train}} = 0.5$, the proportion of responses in the test bank that contain the code is 0.5 and $\overline{E_C} \approx 0.5$). (b) Standard deviation of E_C as $\overline{E_C}$ varies empirically (code: L, $n = 100$, $p_{\text{train}} = 0.5$).

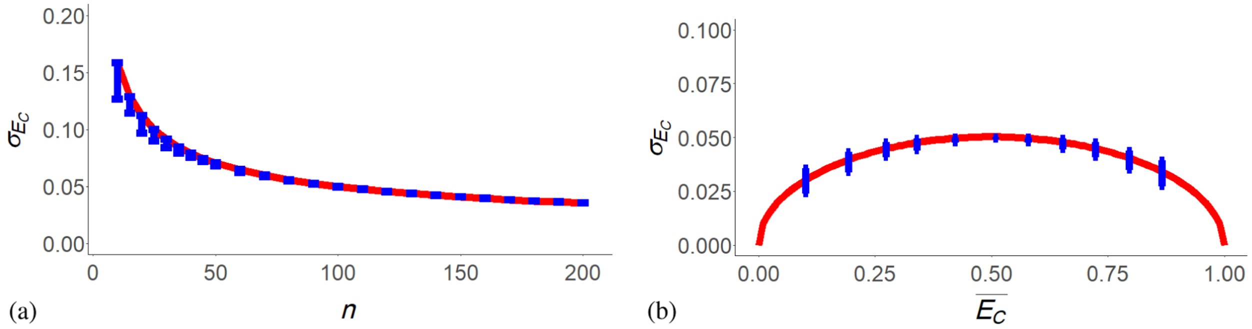

FIG. 11. Accuracy of the approximation $\sqrt{\frac{\overline{E_C}(1-\overline{E_C})}{n}} \approx \sqrt{\frac{(E_C(1-E_C)}{n}}$. Blue error bars show 95% interval of the approximation out of the 1000 samples.

We train an algorithm for the CR code using a training set of size 600 with $p_{\text{train}} = 0.5$. We use 1000 test set samples for each value of n and $\overline{E_C}$, such that each test set sample can be used to make an individual measurement E_C. We compute $\sigma_{E_C} \approx \sqrt{\frac{E_C(1-E_C)}{n}}$ for each individual E_C value from each test set sample. For each value of n and $\overline{E_C}$ we compute the 95% confidence interval for the 1000 values and display it in blue to demonstrate the deviation from $\sqrt{\frac{\overline{E_C}(1-\overline{E_C})}{n}}$ (in red).

To evaluate the quality of the estimate as n varies, we test samples of size n, where n varies between 10 and 200. For each value of n, we draw 1000 test set samples of size n from a test bank with 50% of the responses containing the code. This corresponds to $\overline{E_C} \approx 0.5$. We compute E_C and $\sqrt{\frac{E_C(1-E_C)}{n}}$ for each test set sample and then calculate $\sqrt{\frac{\overline{E_C}(1-\overline{E_C})}{n}}$ at each value of n. The results are plotted in Fig. 11(a). Although there is moderately large deviation for $n < 30$, for most values of n the size of the difference is equal to or less than a rounding error if the statistical uncertainty is reported with 1–2 digits.

We similarly evaluate the quality of the estimate as $\overline{E_C}$ varies. We test values of $\overline{E_C}$ between 0 and 1. From the test data, we create 11 different test banks where the proportion of responses containing the code (as determined by the human coder) ranges from 0 to 1 inclusive in steps of 0.1. For each value of $\overline{E_C}$ (each test bank), we draw 1000 test set samples of size $n = 1000$. We compute E_C and $\sqrt{\frac{E_C(1-E_C)}{n}}$ for each test set sample and then calculate $\sqrt{\frac{\overline{E_C}(1-\overline{E_C})}{n}}$ at each value of $\overline{E_C}$. The results are plotted in Fig. 11(b). For most values of $\overline{E_C}$ included in the plot, the size of the difference is equal to or less than a rounding error if the statistical uncertainty is reported with 1–2 digits. There are a few areas where the deviation is a bit larger, in particular where $\overline{E_C} < 0.2$ or $\overline{E_C} > 0.8$.

3. Independence of statistical uncertainty with training set size

Here we demonstrate that the statistical uncertainty σ_{E_C} is independent of the training set size. We separately train algorithms for the CR code and the L code using $p_{\text{train}} = 0.5$ and training set sizes N_{train} ranging from 100 to 600 inclusive in steps of 50. For each value of N_{train}, we draw 100 test set samples of size $n = 100$ from a test bank of 200 responses with the proportion of responses in the test bank that contain the code fixed at 0.5. We then calculate σ_{E_C} for each value of N_{train}.

Figure 12 demonstrates that σ_{E_C} does not vary based on N_{train} for either the (a) CR code or (b) L code. This result provides evidence that statistical uncertainty is an effect of

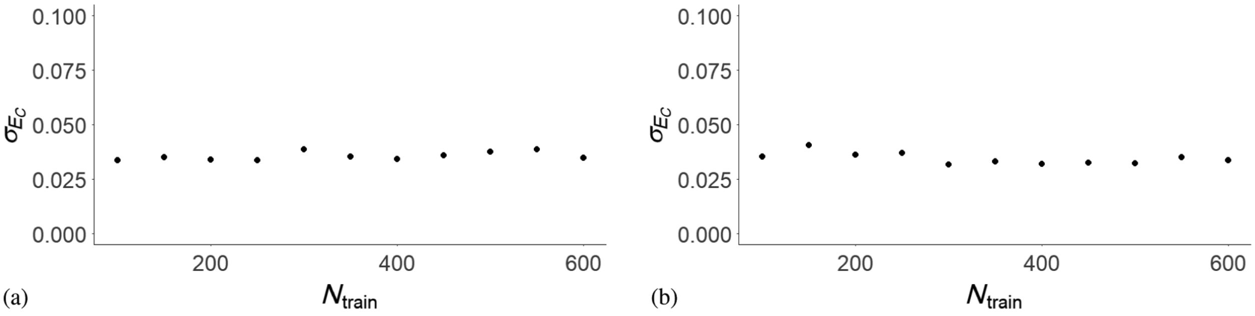

FIG. 12. Statistical uncertainty is independent of the size of the training set. (a) Consistent results code. (b) Limitations code. $n = 100$, $p_{\text{train}} = 0.5$, the proportion of responses in the test bank that contain the code is 0.5, $N_{\text{bank}} = 200$.

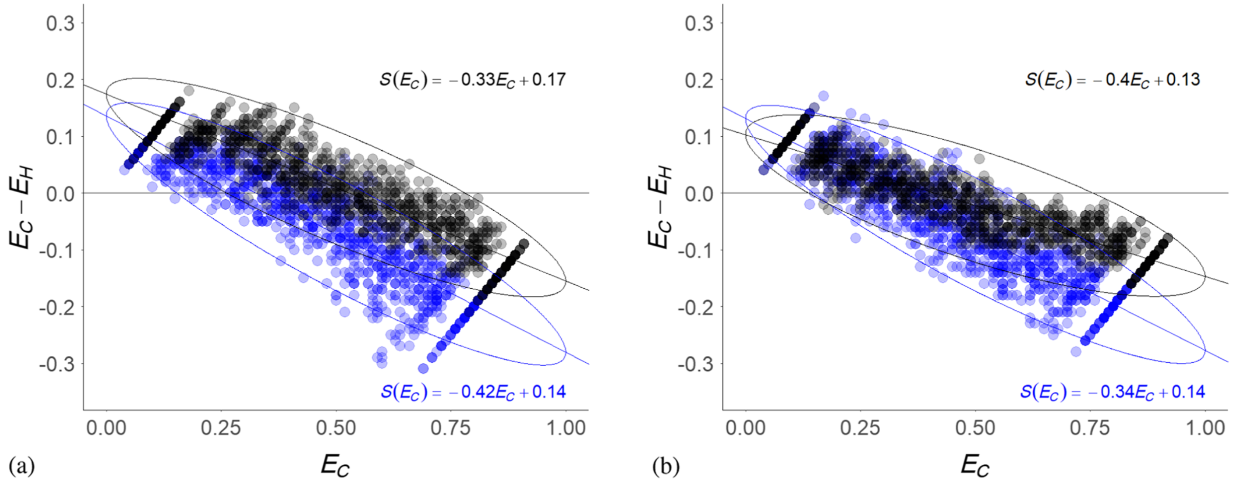

FIG. 13. Given a value of E_C, we can compute the systematic uncertainty using the line of best fit $S(E_C)$. $S(E_C)$ is specific to a training set. When the training data changes, with all else staying constant, $S(E_C)$ changes. Scatter around line of best fit is a result of statistical uncertainty as less than 95% of the data falls within the black oval depicting $2 \times \sigma_{E_C}$ [see Eq. (3)]. (a) $S(E_C)$ for CR code with a different training set (blue) along with $S(E_C)$ from Fig. 5(a) (black). (Code: CR, $p_{\text{train}} = 0.5$) (b) $S(E_C)$ for L code with a different training set (blue) along with $S(E_C)$ from Fig. 5(b) (black) (Code: L, $p_{\text{train}} = 0.5$).

the limited sample size rather than the size of the training set (a proxy for the quality of the trained model).

APPENDIX C: EFFECT OF DIFFERENT TRAINING SETS ON MODELS OF SYSTEMATIC UNCERTAINTY

In the main text, we provide evidence that the systematic difference between human and machine coders can be modeled as a linear function $S(E_C)$. Here, we provide evidence that the specific function is dependent on the trained model. That is, if the training data used to generate a trained model are changed, $S(E_C)$ changes. This analysis informs our recommendation to generate $S(E_C)$ for *a single trained model* that will be applied to new data.

We sample a training set of size 600 for the CR code (different from the training set used in Fig. 5), both of which are still balanced (50% of responses contain the code). From the data not included in this training set, we generate a set of nine different test banks where the proportion of responses containing the code varies from 0.1 to 0.9 inclusive. The size of each test bank is $N_{\text{bank}} = 300$. For each test bank, we compute E_C and E_H for 100 sample test sets of size $n = 100$ pulled from the test bank. In Fig. 13(a), we plot $E_C - E_H$ vs E_C for this new data (in blue) alongside the data from Fig. 5 (in black). We repeat this same process for the L code (panel b).

For the CR code, the difference between the systematic uncertainty for the two different trained models extends beyond statistical uncertainty throughout the range of E_C. For instance, for the CR code $S(E_C = 0.25) = 0.088$ for the original trained model in Fig. 5, but $S(E_C = 0.25) = 0.035$ when using the different training set shown in Fig. 13. The statistical uncertainty where $E_C = 0.25$ is $\sigma_{E_C}(0.25) = 0.043$; thus, the difference in these two estimate, $0.088 - 0.035 = 0.053$, is slightly larger than $\sigma_{E_C}(0.25)$. The difference grows as E_C increases. When $E_C = 0.75$, for example, $S(E_C = 0.75) = -0.077$ for the original trained model and $S(E_C = 0.75) = -0.175$ for the different training set. This difference, $-0.077 - (-0.175) = 0.098$, is more than twice the statistical uncertainty $\sigma_{E_C}(0.75) = 0.043$.

For the L code, the difference between the systematic uncertainty for two different trained models is indistinguishable for $E_C < 0.5$ but extends beyond the statistical uncertainty at high values of E_C. Thus, changing the trained model may be beyond statistical uncertainty, especially at extreme values of E_C. We expect that this occurs because each trained model learns slightly different coding rules, even though both training sets are balanced.

We note that in this analysis we are comparing only two specific examples of trained models for each code, thus the comparison above cannot be generalized to other examples. For instance, we cannot conclude that a third example of a trained model for the L code is more likely to be indistinguishable from our existing examples than a third example of a trained model for the CR code.

[1] S. Kanim and X. C. Cid, Demographics of physics education research, Phys. Rev. Phys. Educ. Res. **16**, 020106 (2020).

[2] J. W. Burton, M. Stein, and T. B. Jensen, A systematic review of algorithm aversion in augmented decision making, J. Behav. Decis. Making **33**, 220 (2020).

[3] B. J. Dietvorst, J. P. Simmons, and C. Massey, Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them, Manage. Sci. **64**, 1155 (2018).

[4] R. Misra and J. Grover, *Sculpting Data for ML: The First Act of Machine Learning* (Misra, Rishabh and Grover, Jigyasa, 2021).

[5] R. Misra, News category dataset, arXiv:2209.11429.

[6] L. Ding, Theoretical perspectives of quantitative physics education research, Phys. Rev. Phys. Educ. Res. **15**, 020101 (2019).

[7] B. Sherin, A computational study of commonsense science: An exploration in the automated analysis of clinical interview data, J. Learn. Sci. **22**, 600 (2013).

[8] B. Sherin, N. B. Kersting, and M. Berland, Learning analytics in support of qualitative analysis, in *Proceedings of International Conference of the Learning Sciences, ICLS, 1* (London, United Kingdom, 2018), pp. 464–471.

[9] P. Hur, N. Machaka, C. Krist, and N. Bosch, Informing expert feature engineering through automated approaches: Implications for coding qualitative classroom video data, in *LAK23: 13th International Learning Analytics and Knowledge Conference*, LAK2023 (Association for Computing Machinery, New York, NY, 2023), pp. 630–636.

[10] M. Kubsch, C. Krist, and J. M. Rosenberg, Distributing epistemic functions and tasks: A framework for augmenting human analytic power with machine learning in science education research, J. Res. Sci. Teach. **60**, 423 (2023).

[11] S. Mariegaard, L. D. Seidelin, and J. Bruun, Identification of positions in literature using thematic network analysis: The case of early childhood inquiry-based science education, Int. J. Research Method Educ. **45**, 518 (2022).

[12] J. M. Rosenberg and C. Krist, Combining machine learning and qualitative methods to elaborate students' ideas about the generality of their model-based explanations, J. Sci. Educ. Technol. **30**, 255 (2021).

[13] L. K. Nelson, Computational grounded theory: A methodological framework, Sociol. Methods Res. **49**, 3 (2020).

[14] P. Tschisgale, P. Wulff, and M. Kubsch, Integrating artificial intelligence-based methods into qualitative research in physics education research: A case for computational grounded theory, Phys. Rev. Phys. Educ. Res. **19**, 020123 (2023).

[15] J. Wilson, B. Pollard, J. M. Aiken, M. D. Caballero, and H. J. Lewandowski, Classification of open-ended responses to a research-based assessment using natural language processing, Phys. Rev. Phys. Educ. Res. **18**, 010141 (2022).

[16] C. M. Nakamura, S. K. Murphy, M. G. Christel, S. M. Stevens, and D. A. Zollman, Automated analysis of short responses in an interactive synthetic tutoring system for introductory physics, Phys. Rev. Phys. Educ. Res. **12**, 010122 (2016).

[17] R. Fussell, A. Mazrui, and N. G. Holmes, Machine learning for automated content analysis: Characteristics of training data impact reliability, presented at PER Conf. 2022, Grand Rapids, MI, 10.1119/perc.2022.pr.Fussell.

[18] J. Campbell, K. Ansell, and T. Stelzer, Using IBM's Watson to automatically evaluate student short answer responses, presented at PER Conf. 2022, Grand Rapids, MI, 10.1119/perc.2022.pr.Campbell.

[19] T. Ullmann, Automated analysis of reflection in writing: Validating machine learning approaches, Intl. J. Artif. Intell. Educ. **29**, 217 (2019).

[20] P. Wulff, D. Buschhuter, A. Westphal, A. Nowak, L. Becker, H. Robalino, M. Stede, and A. Borowski, Computer-based classification of preservice physics teachers' written reflections, J. Sci. Educ. Technol. **30**, 1 (2021).

[21] M. Thomas, S. Bagley, and M. Urban-Lurain, Using machine learning algorithms to categorize free responses to calculus questions, in *Proceedings of the Twenty-first Annual Conference on Research in Undergraduate Mathematics Education* (The Special Interest Group of the Mathematical Association of America (SIGMAA) for Research in Undergraduate Mathematics Education, San Diego, CA, 2018).

[22] R. Jiang, J. Gouvea, D. Hammer, and S. Aeron, Automatic coding of students' writing via contrastive representation learning in the wasserstein space, arXiv:2011.13384.

[23] L. K. Nelson, D. Burk, M. Knudsen, and L. McCall, The future of coding: A comparison of hand-coding and three types of computer-assisted text analysis methods, Sociol. Methods Res. **50**, 202 (2021).

[24] W. van Atteveldt, M. A. C. G. van der Velden, and M. Boukes, The validity of sentiment analysis: Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms, Commun. Methods. Meas. **15**, 121 (2021).

[25] T. Odden, A. Marin, and M. D. Caballero, Thematic analysis of 18 years of physics education research conference proceedings using natural language processing, Phys. Rev. Phys. Educ. Res. **16**, 010142 (2020).

[26] T. Odden, A. Marin, and J. L. Rudolph, How has science education changed over the last 100 years? an analysis using natural language processing, Sci. Educ. **105**, 653 (2021).

[27] J. M. Geiger, L. M. Goodhew, and T. O. B. Odden, Developing a natural language processing approach for analyzing student ideas in calculus-based introductory physics, presented at PER Conf. 2022, Developing a natural language processing approach for analyzing student ideas in calculus-based introductory physics, Grand Rapids, MI, 10.1119/perc.2022.pr.Geiger.

[28] A. Bralin, J. Morphew, C. Rebello, and N. S. Rebello, Analysis of student essays in an introductory physics course using natural language processing, presented at PER Conf. 2023, Sacramento, CA, 10.1119/perc.2023.pr.Bralin.

[29] J. R. Landis and G. G. Koch, The measurement of observer agreement for categorical data, Biometrics **33**, 159 (1977).

[30] Y. Wang and M. I. Jordan, Desiderata for representation learning: A causal perspective, arXiv:2109.03795.

[31] M. Srivastava, T. B. Hashimoto, and P. Liang, Robustness to spurious correlations via human annotations, arXiv:2007.06661.

[32] M. Hu, Z. Zhang, S. Zhao, M. Huang, and B. Wu, Uncertainty in natural language processing: Sources, quantification, and applications, Uncertainty in natural language processing: Sources, quantification, and applications, arXiv:2306.04459.

[33] T. O. Odden, R. K. Fussell, C. Green, and N. T. Young, Machine learning methods in PER: Intuition and methodological discussion, in *Parallel Session at the Physics Education Research Conference 2022, Grand Rapids, MI* (2022), https://www.per-central.org/perc/2022/detail.cfm?ID=8672.

[34] V. Adlakha and E. Kuo, Critical issues in statistical causal inference for observational physics education research, Phys. Rev. Phys. Educ. Res. **19**, 020160 (2023).

[35] BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, Evaluation of measurement data - guide to the expression of uncertainty in measurement, Joint Committee for Guides in Metrology, JCGM 100 (2008).

[36] D. Hu and B. M. Zwickl, Examining students' views about validity of experiments: From introductory to Ph.D. students, Phys. Rev. Phys. Educ. Res. **14**, 010121 (2018).

[37] E. M. Stump, M. Dew, G. Passante, and N. G. Holmes, Context affects student thinking about sources of uncertainty in classical and quantum mechanics, Phys. Rev. Phys. Educ. Res. **19**, 020157 (2023).

[38] E. M. Stump, M. Hughes, G. Passante, and N. G. Holmes, Comparing introductory and beyond-introductory students' reasoning about uncertainty, Phys. Rev. Phys. Educ. Res. **19**, 020147 (2023).

[39] R. Rifkin and A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. **5**, 101 (2004), https://www.jmlr.org/papers/volume5/rifkin04a/rifkin04a.pdf.

[40] S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python* (O'Reilly Media Inc., Newton, MA, 2009).

[41] K. Sparck-Jones, A statistical interpretation of term specificity and its application in retrieval, J. Documentation **28**, 11 (1972), https://www.emerald.com/insight/content/doi/10.1108/eb026526/full/pdf?title=a-statistical-interpretation-of-term-specificity-and-its-application-in-retrieval.

[42] J. Brownlee, *Deep Learning for Natural Language Processing*, v1.9 ed. (Jason Brownlee Books, 2021), p. 108.

[43] S. Sarica and J. Luo, Stopwords in technical language processing, PLoS One **16**, e0254937 (2021).

[44] D. Tannen, *Framing in Discourse* (Oxford University Press, Oxford, U.K., 1993).

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011), https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[46] F. Chollet *et al.*, Keras, https://keras.io (2015).

[47] J. Pennington, R. Socher, and C. D. Manning, Glove: Global vectors for word representation, in *Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1532–1543.

[48] J. M. Aiken, R. DeBin, H. J. Lewandowski, and M. D. Caballero, Framework for evaluating statistical models in physics education research, Phys. Rev. Phys. Educ. Res. **17**, 020104 (2021).

[49] A. Vabalas, E. Gowen, E. Poliakoff, and A. Casson, Machine learning algorithm validation with a limited sample size, PLoS One **14**, e0224365 (2019).

[50] N. T. Young and M. D. Caballero, Predictive and explanatory models might miss informative features in educational data, J. Educ. Data Mining **13**, 31 (2021).

[51] J. D. Bransford and D. L. Schwartz, Rethinking transfer: A simple proposal with multiple implications, Rev. Res. Educ. **24**, 61 (1999).

[52] R. J. Shumway, Negative instances and mathematical concept formation: A preliminary study, J. Res. Math. Educ. **2**, 218 (1971).

[53] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explor. Newsl. **6**, 20 (2004).

[54] R. K. Fussell, NLP uncertainty (2023), https://github.com/rkfussell/NLP_uncertainty.

[55] Z. Chen, E. Frederick, C. Cui, M. Khan, C. Klatt, M. Huang, and S. Su, Reforming physics exams using openly accessible large isomorphic problem banks created with the assistance of generative AI: An explorative study, arXiv:2310.14498.

[56] W. Liang, M. Yuksekgonul, Y. Mao, E. Wu, and J. Zou, GPT detectors are biased against non-native English writers, Patterns **4**, 100779 (2023).