Scaling Up Writing in the Curriculum: Batch Mode Active Learning for Automated Essay Scoring

Scott Hellman, Mark Rosenstein, Andrew Gorman, William Murray, Lee Becker, Alok Baikadi, Jill Budden, Peter W. Foltz
Pearson
Boulder, CO
{scott.hellman, mark.rosenstein, andrew.gorman, william.murray, lee.becker, alok.baikadi, jill.budden, peter.foltz}@pearson.com

ABSTRACT
Automated essay scoring (AES) allows writing to be assigned in large courses and can provide instant formative feedback to students. However, creating models for AES can be costly, requiring the collection and human scoring of hundreds of essays. We have developed and are piloting a web-based tool that allows instructors to incrementally score responses to enable AES scoring while minimizing the number of essays the instructors must score. Previous work has shown that techniques from the machine learning subfield of active learning can reduce the amount of training data required to create effective AES models. We extend those results to a less idealized scenario: one driven by the instructor's need to score sets of essays, in which the model is trained iteratively using batch mode active learning. We propose a novel approach inspired by a class of topological methods, but with reduced computational requirements, which we refer to as topological maxima. Using actual student data, we show that batch mode active learning is a practical approach to training AES models. Finally, we discuss implications of using this technology for automated customized scoring of writing across the curriculum.

ACM Classification Keywords
I.1.2 Algorithms; F.1.2 Modes of Computation: Online computation; I.2.7 Natural Language Processing: Text analysis; K.3 Computers and Education

Author Keywords
Active learning; Automated Essay Scoring; Machine Learning; Topology

INTRODUCTION
The ability to express yourself through writing – the ability to petition, to generate cogent arguments, and to express ideas with clarity and vigor – is a critical life skill. It is crucial for success in school, in career, and as a fully functional citizen. It is a skill that develops with practice (e.g., [14]), but assessing writing to support that practice, especially early drafts, is a time-intensive human task. Over the last 20 years, Automated Essay Scoring (AES) has steadily improved, so that, for many writing tasks, automated assessment performance is comparable to that achieved by human scoring [32, 33]. Most automated scoring is built upon machine learning models trained on essay sets that have been scored by multiple human scorers. In this supervised learning paradigm, the models become able to mimic the human scoring. Many of these models are trained for high stakes examinations, which requires an intensive and expensive process with highly trained scorers, and extensive quality assurance. In situations with such high impact, this effort can be justified, but our interest is at the formative practice stage of the student writing experience for large courses.

We envision writing being integral to instruction. Writing should be part and parcel of taking a class; not a rare event, but the norm. This vision requires reducing the effort involved in creating and scoring writing prompts and in developing their associated automated assessment models. We have developed and are piloting a web-based software system that supports the entire student writing process: instructor prompt and rubric creation, student essay writing, human scoring, model building, and archiving of models for reuse [3]. Automated scoring is supported in the system when students write responses to a prompt. If scoring models already exist for the prompt, students may receive immediate feedback when they submit drafts of their essays. If a model does not exist, the essays are batched at the due date and directed to humans for scoring. These human scorers could be the instructor, one or more teaching assistants, or professional scorers. After the essays are scored, those scores are returned to the students. In the background, a machine learning system uses the scores in an attempt to build an automated scoring model.

The overwhelming need for writing in the classroom originates from it being an authentic task and a lifetime skill. It is also a constructed response that requires complex thinking skills, which allows for an expansive range of correct and incorrect executions. Beginning with the earliest attempts to standardize the scoring of essays in the 1960s [9], it was recognized that

L@S '19, June 24–25, 2019, Chicago, IL, USA

 ISBN 978-1-4503-6804-9/19/06…$15.00
DOI: https://doi.org/10.1145/3330430.3333629

the boundaries between essay scores were fuzzy. In high stakes scoring, even well trained human scorers typically disagree, often substantially. Beyond training and monitoring scorers, this issue is mitigated in high stakes scoring by requiring two scorers (in the modern world, one of those scorers is often automated). If the two scorers disagree sufficiently, a third scorer is brought in to resolve the disagreement. In addition to this innate imprecision, human scorers also often exhibit flaws in their scoring, which result from rater bias [37]. In modern high stakes scoring, the expected inter-rater correlation is in the range of 0.7–0.8, and those historical human limitations have become enshrined as a standard by which automated scoring is currently judged [36]. For formative writing, we are attempting to provide useful and accurate feedback while also attempting to relax some of the high stakes constraints (such as multiple professional scorers) that would make writing in the classroom impractical.

The current process for building automated scoring models for a writing prompt involves gathering a fixed number of student essays (usually a minimum of 300 or more), human scoring them all, building a model, and then determining if the model is accurate (e.g., agrees well with the human scorers). If the number of essays is not sufficient, then another essay collection period is initiated. An essential step to decreasing human scoring effort is decreasing the required number of scored essays. In simulations that we have run, it is often possible to build a satisfactory model with far fewer than 300 scored essays. In this work, we investigate an alternative model building process that, by determining which essays would most benefit the training of the model, reduces the total number of scored essays required.

In this approach, a small subset of essays are selected for scoring. After that subset is scored by a human, a model is built and evaluated. If the model meets performance criteria, the remainder of the essays can be automatically scored, and the model archived for future use on the prompt. If the model is not acceptable, the process is repeated until either a satisfactory model is obtained or it is determined that the prompt is not a good candidate for autoscoring. By selecting the essays that are most relevant to model training, this approach has the potential to substantially reduce the human scoring workload.

Active learning [30] is a subfield of machine learning concerned with reducing the amount of labeled data needed to train a supervised model. This is of particular interest when there is a large unlabeled dataset and acquiring labels is expensive. This exactly describes our situation, where we have a set of unscored student essays and we want to train a satisfactory model while scoring as few essays as possible.

In this work, we describe active learning, and discuss active learning techniques that are especially promising for use in selecting batches of essays. We propose a novel simplification of a topological active learning approach, which we call *topological maxima*. We place this research in the spectrum of active learning applications, from more general natural language processing (NLP) to its use in assessment. We then apply these techniques on the ASAP (Automated Student Assessment Prize) essay data set[1] and on essays from a set of prompts that are currently being used in college level courses. Finally, we conclude with some observations on the techniques, describe implications for delivering automated writing scoring at scale, and note future directions.

ACTIVE LEARNING

We are concerned with the active learning paradigm of pool-based sampling [30]. Pool-based sampling assumes that we have a single static set U of unlabeled datapoints. Given access to an oracle (such as a human scorer) that labels datapoints at some cost, the goal is to construct a set of labeled datapoints L that allows us to train a performant supervised model while minimizing the total labeling cost.

Many active learning techniques are sequential – that is, they iteratively select a single datapoint from U, query the oracle for the label, and add that point to L. The downside of sequential approaches is that the human labeler must wait while the active learning algorithm selects the next point for labeling. This downtime can be substantial, as many variants of active learning require access to a model trained on the current set L, forcing the human to wait for both active learning and model training. For example, in our current implementation, it can take up to 10 minutes to generate, evaluate, and deploy a new model. For this reason, we are interested in selecting subsets of U for labeling, in what is known as batch mode active learning. By using batch mode active learning, a scorer can score multiple essays before waiting for model retraining.

Batch mode active learning generally performs worse than sequential active learning, but has been shown to be effective in contexts where downtime for the human labelers is expensive [22]. Some active learning algorithms do not rely on the labels to generate their selections, and for these algorithms, operating in sequential or batch mode results in the same selections. However, for uncertainty sampling-based methods that rely on computing the current model's uncertainty, the difference between sequential and batch mode active learning can be substantial [2]. For some types of supervised models, the problem of identifying sets of points that minimize uncertainty can be cast as a submodular function optimization problem, allowing for algorithms that perform well with good theoretical guarantees [7, 19, 16, 8].

For methods that utilize the supervised model's uncertainty, there is a cold start problem: when selecting the very first batch, there is no existing trained model to estimate the uncertainty with. One way to mitigate this is to select the initial batch (referred to as a seed set) using a different algorithm. Seed set selection can have a substantial impact on overall performance [10].

In natural language processing applications, active learning has been shown to be effective in a variety of tasks, such as named entity recognition [35, 31], annotation noise detection [27], sentiment classification [23], word sense disambiguation [10], and text categorization [15].

[1] https://www.kaggle.com/c/ASAP-AES

Active learning has found applications in educational NLP tasks as well. Niraula and Rus showed that active learning can be used when training classifiers for the evaluation of automatically generated gap-fill questions [25]. Horbach and Palmer demonstrated that batch mode active learning can provide benefits when training short-answer scoring models [17]. In particular, they find that uncertainty sampling-based approaches generally perform better than random sampling, which is a commonly used baseline in the evaluation of active learning algorithms.

To the best of our knowledge, the only prior work that investigates active learning for AES is Dronen et al., in which the authors demonstrate that active learning reduces the amount of training data needed to train an ordinary least-squares regression model for AES [11]. In their active learning experiments, they evaluate model performance with respect to training set size. However, they do not build these training sets incrementally – at each training set size, active learning is used to select an entirely new training set. In contrast, we pose the problem of training an AES model as a batch mode active learning task, and our training sets are incrementally constructed from those batches. Our approach is designed to be a realistic simulation of active learning used by teachers in large courses, and explicitly accounts for the fact that we are selecting batches of essays simultaneously.

BATCH MODE ACTIVE LEARNING

In this section, we describe the active learning algorithms that we use in our experiments. We approach AES as a regression task. Student essays are passed through a natural language processing pipeline which extracts various linguistic features, allowing each essay to be represented by a vector of real numbers denoted by $\mathbf{x}$. Random forests are used as our supervised modeling technique, and so we have selected active learning algorithms that can perform well in this setting. We use random forests because they have been found to perform well in real-world AES applications [13].

For approaches that need the uncertainty of the model at a point $\mathbf{x}$, we use the variance of the individual trees' predictions for $\mathbf{x}$, and denote this variance as $v(\mathbf{x})$. For approaches that require a distance metric, we denote the distance by d. We assume that the size of the selected batch is a user-defined constant, denoted by b.

Kennard-Stone

The *Kennard-Stone* active learning algorithm attempts to uniformly sample points from the feature space [18]. Kennard-Stone begins by selecting the two points satisfying

$$\operatorname*{argmax}_{x,x' \in U} d(x,x'),$$

that is, the two points in U that are furthest apart from each other. All further points are selected to maximize the minimum distance to any point in L:

$$\operatorname*{argmax}_{x \in U} \left(\min_{x' \in L} d(x,x') \right).$$

Kennard-Stone is deterministic and its behavior is identical in both sequential and batch mode settings. Our use of Kennard-Stone is based on findings in Dronen et al. that it performed well for AES model training [11].

Uncertainty Sampling

Uncertainty sampling [21] is a popular and effective active learning approach. Uncertainty sampling selects the point in U that the model is most uncertain about:

$$\operatorname*{argmax}_{x \in U} v(x).$$

Since we are using random forests, this can also be viewed as query by committee for regression [5]. Intuitively, if the trees have substantially differing predictions for a specific point, then it is likely an important point to have labeled by the oracle.

When used for batch mode active learning, uncertainty sampling selects the b highest variance points. However, uncertainty sampling can perform poorly in batch contexts, because the points selected may be too similar to each other [30]. For example, if the two points with the highest variance are identical, then batch mode uncertainty sampling will select both, even though selecting only one would provide the model with almost as much information about that region of the feature space.

Bounded Coordinated Matching

To address the issues of uncertainty sampling in batch mode active learning, Azimi et al. introduced the *bounded coordinated matching* (BCM) algorithm [2]. BCM assumes that the selections made by sequential uncertainty sampling are generated by a k-Matching Mixture Model (k-MM model). The k-MM model is similar to a Gaussian Mixture Model, but rather than assuming that the underlying generative process is i.i.d, k-MM models generate k points by sampling one point from each of the k components.

Parameter estimation for the k-MM model is done by Monte Carlo simulation. For each of the N simulations, we select k points using simulated sequential uncertainty sampling. We simulate sequential uncertainty sampling by iteratively selecting the highest variance point $\mathbf{x}$, labeling it by sampling a label y from the model's posterior distribution for $\mathbf{x}$, adding $(\mathbf{x}, y)$ to L, and retraining the model. As we are using random forests for regression, we sample from the posterior distribution by randomly selecting one tree in the ensemble and using its prediction for $\mathbf{x}$ as y.

Each of these N simulations generates a selection of k points; for the i-th simulation, we denote this selection as S_i. Using these simulated selections, BCM is able to approximate the mean vectors μ of the underlying k-MM model. Due to the nature of the task, these mean vectors are themselves points in U, and so by taking $k = b$, we can use μ as the batch selected by BCM. That is, BCM attempts to select the k points in U that best represent the repeated simulations of uncertainty sampling.

BCM approximates μ as follows. Assuming that we have a set of candidate mean vectors $\hat{\mu}$, for the selections S_i of the i-th

Monte Carlo simulation, we can compute the earth mover's distance [28] $D(\hat{\mu}, S_i)$ between those selections and the mean vectors (with our distance metric d as the ground distance used in the computation of D). BCM seeks to minimize the objective function

$$g(\hat{\mu}) = \sum_{i=1}^{N} D(\hat{\mu}, S_i)$$

by beginning with $\hat{\mu} = U$ and then greedily removing points from $\hat{\mu}$ until $|\hat{\mu}| = k$. Because g is supermodular, the error of this approximation has well-defined bounds [2].

Azimi et al. provide a number of techniques to speed up this computation [2]. However, due to the Monte Carlo simulations and repeated earth mover's distance computations, BCM is much more computationally intensive than the other approaches considered in this work.

Maximum Persistence

Introduced in Maljovec et al., *maximum persistence* adapts uncertainty sampling to batch mode active learning by interpreting the task as a topological problem [24]. Rather than selecting the points with the highest variance, maximum persistence selects the points where the variance function is most *persistent*.

Persistence is a way of quantifying how robust topological features are to noise. In this case, the topological features that we are interested in are the maxima of the variance function v. Intuitively, the persistence of a maximum $\mathbf{m}$ indicates how much the value of v could increase at points near $\mathbf{m}$ before $\mathbf{m}$ ceases to be a maximum.

To compute persistence, we first need to formalize a notion of which points are near each other. To do so, we create a neighbor graph G from U under our distance metric d. In this work, we use k-nearest-neighbor graphs, but any graph that captures the spatial relationships of the points in U could work. We denote the vertices in G by G_V, and the edges by G_E.

As we are specifically interested in the maxima of the variance v over G, we construct a discrete approximation of the gradient of v. For each vertex $\mathbf{x}_i \in G_V$, we can build a set

$$\Delta_i = \{\delta_{ij} \mid e_{ij} \in E(\mathbf{x}_i)\},$$

where $E(\mathbf{x}_i)$ is the set of edges incident to $\mathbf{x}_i$, e_{ij} is the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$, and

$$\delta_{ij} = \frac{v(\mathbf{x}_j) - v(\mathbf{x}_i)}{d(\mathbf{x}_i, \mathbf{x}_j)}.$$

Δ_i is an approximation of the derivative of v at $\mathbf{x}_i$ in every possible direction (that is, to every vertex that $\mathbf{x}_i$ is connected to).

We define $\mathbf{x}_i$ to be a maximum when none of its adjacent vertices have higher variance, that is, when

$$\forall \delta_{ij} \in \Delta_i, \delta_{ij} < 0.$$

Given these maxima, we can use the algorithm in [6] to compute the persistence; we briefly outline the algorithm here. We first assign each vertex $\mathbf{x_i}$ to a maximum $\mathbf{m}_i$. If $\mathbf{x}_i$ is a maximum, we assign it to itself. Otherwise, we find the neighbor $\mathbf{x}_j$ of steepest ascent (the neighbor $\mathbf{x}_j$ that corresponds to $\text{argmax}_j(\Delta_i)$), recursively assign $\mathbf{x}_j$ to a maximum $\mathbf{m}_j$, and then assign $\mathbf{x}_i$ to $\mathbf{m}_j$ as well. We refer to the set of points assigned to a maximum $\mathbf{m}$ as the *descending manifold* for $\mathbf{m}$.

Once we have these manifolds, we begin constructing a new graph $\hat{G}$ by iteratively inserting every point $\mathbf{x}_i \in G_V$ into $\hat{G}$ in descending order of their values $v(\mathbf{x}_i)$. For each vertex $\mathbf{x}_i$ added to $\hat{G}$, we also add an edge to every other vertex $\mathbf{x}_j$ currently in $\hat{G}_V$ for which $e_{ij} \in G_E$. During this process, whenever at least one other vertex $\mathbf{x}_j$ connected to $\mathbf{x}_i$ belongs to a different manifold than $\mathbf{x}_i$, we have connected two descending manifolds.

When this occurs, we merge the two manifolds and compute persistence. We assume that $v(\mathbf{m}_i) > v(\mathbf{m}_j)$. We merge the two manifolds by declaring that both now correspond to $\mathbf{m}_i$, the higher of the two maxima. The persistence of $\mathbf{m}_j$ is then defined to be $v(\mathbf{m}_j) - v(\mathbf{x}_i)$. This value quantifies how far down the descending manifold of $\mathbf{m}_j$ we have to travel before we encounter a manifold belonging to a higher maxima.

After all vertices have been added to $\hat{G}$, only one manifold will remain. We define that the maximum associated with this manifold has infinite persistence.

Once we have computed the persistence of the maxima, we then select the b maxima with the highest persistence. If not enough maxima exist, we fall back to uncertainty sampling to select the remaining points.

Topological Maxima

We also consider a simplification of maximum persistence, which was briefly discussed in [24], but to the best of our knowledge has never been empirically tested. This simplification leverages the neighbor graph, but does not compute persistence, allowing us to determine how important persistence is to maximum persistence's ability to select useful batches. As when computing maximum persistence, we compute a neighbor graph, label each vertex with its variance, and find the maxima of the graph. Rather than computing the persistence, however, we stop here and select the b maxima with the highest variance. As in maximum persistence, if not enough maxima exist, we fall back to uncertainty sampling to select the remaining points. We refer to this algorithm as *topological maxima*.

DATASETS

We evaluate these active learning approaches on two datasets. The first is a set of seven college-level formative writing prompts: five psychology prompts and two accounting prompts. These prompts are currently in use in college courses. Each prompt has a five-trait rubric with scores ranging from 1 to 4. We are primarily interested in scoring for content, so in this work, we focus solely on one rubric trait – the content score. Each essay was scored by two trained scorers, and we use the mean human score as the true score of each essay. While we are ultimately interested in situations where there is only one scorer (the instructor), for evaluating our algorithms

Prompt	Type	Grade Level	Score Range	Number of Essays	Mean Word Count
Psychology-1	Psychology	College	1–4	343	385
Psychology-2	Psychology	College	1–4	308	473
Psychology-3	Psychology	College	1–4	424	383
Psychology-4	Psychology	College	1–4	448	258
Psychology-5	Psychology	College	1–4	477	263
Accounting-1	Accounting	College	1–4	309	337
Accounting-2	Accounting	College	1–4	349	356
ASAP 1	Persuasive	8	2–12	1783	410
ASAP 2a	Persuasive	10	1–6	1800	430
ASAP 2b	Persuasive	10	1–4	1800	430
ASAP 3	Source	10	0–3	1726	121
ASAP 4	Source	10	0–3	1770	104
ASAP 5	Source	8	0–4	1805	139
ASAP 6	Source	10	0–4	1800	171
ASAP 7	Narrative	7	0–30	1569	189
ASAP 8	Narrative	10	0–60	723	681

Table 1. Characteristics and summary statistics of prompts used in experiments.

we have chosen to use prompts with scores from multiple scorers. Given issues such as rater bias mentioned earlier, using the mean of multiple scores provides a better estimate of the student's true performance on the essay than either individual score would.

We also use the ASAP AES dataset [34], which consists of eight different prompts and student responses to those prompts. These cover three kinds of tasks: persuasive writing, response to source material, and narrative writing. These prompts were assigned to students in grades 7 to 10. Most prompts have only one trait, but prompt 2 has two traits. We model these two traits separately by splitting prompt 2 into two modeling tasks, one for each trait, which we will refer to as prompts 2a and 2b. For prompts 7 and 8, the trait score range is substantially different from the 1 to 4 range in our college-level prompts (0 to 30 and 0 to 60, respectively). We include these prompts in our experiments, but note that different approaches or measures might be more appropriate given these score ranges.

The characteristics of these datasets are summarized in Table 1.

For these active learning experiments we used a total of 5511 features. This feature set is intended to be useful in describing many aspects of an essay, but is specifically designed for scoring content. The features divide naturally into three classes that have frequently been used in automated scoring of writing (e.g. [1, 12, 4]). The first class of features can be described as surface features, and provides metrics on the surface level of the language, such as word counts, mean syllables per word, and mean sentence length. The second class derives from information theoretic statistical features such as n-gram likelihoods [29]. This class of features measures how likely it is that word patterns in a passage would occur in large samples of English text, as well as other regularities in the patterns of language use, giving a measure of how "English-like" the word use is. These features capture such aspects of language as syntactic complexity, flow, and word choice. The third class of features involves content overlap at the word and character level as well as statistics-based semantics measures, based on latent semantic analysis [20].

EXPERIMENTS

We use scikit-learn's random forest implementation [26]. In all experiments, we use forests of 300 trees, with 12 randomly selected features considered at each split. For algorithms that require a distance metric, we use the Euclidean distance. For algorithms that utilize a neighbor graph, we use 4-nearest-neighbor graphs, and for bounded coordinated matching, we use 10 Monte Carlo simulations. These settings were found to perform well in preliminary experiments. We evaluate model performance by computing the Pearson correlation coefficient of the model predictions and the average human score.

Our initial seed set is of size 4, with essays selected using the Kennard-Stone algorithm. We use a batch size of 10, and terminate the run after 15 batches, or when the model achieves 95% of the performance achieved on the full training set, whichever comes later. After each batch is selected, we label the batch with its true scores, and then re-train the random forest.

For each prompt, we create 100 randomized training splits, reserving 10% of the dataset for testing. We then run each active learning algorithm on each split. We use random selection as a baseline. We also compute the model's performance when trained on the entire training set.

DISCUSSION

We summarize our results in Figure 1. This figure shows the average training set size needed to achieve 95% of the performance of a model trained on the full training set. We use this 95% performance threshold to quantify the impact of each active learning approach on the amount of training data needed to train a performant model, while also accounting for the fact that our modeling process is inherently random, and so even two models trained on exactly the same training data would not achieve exactly the same performance.

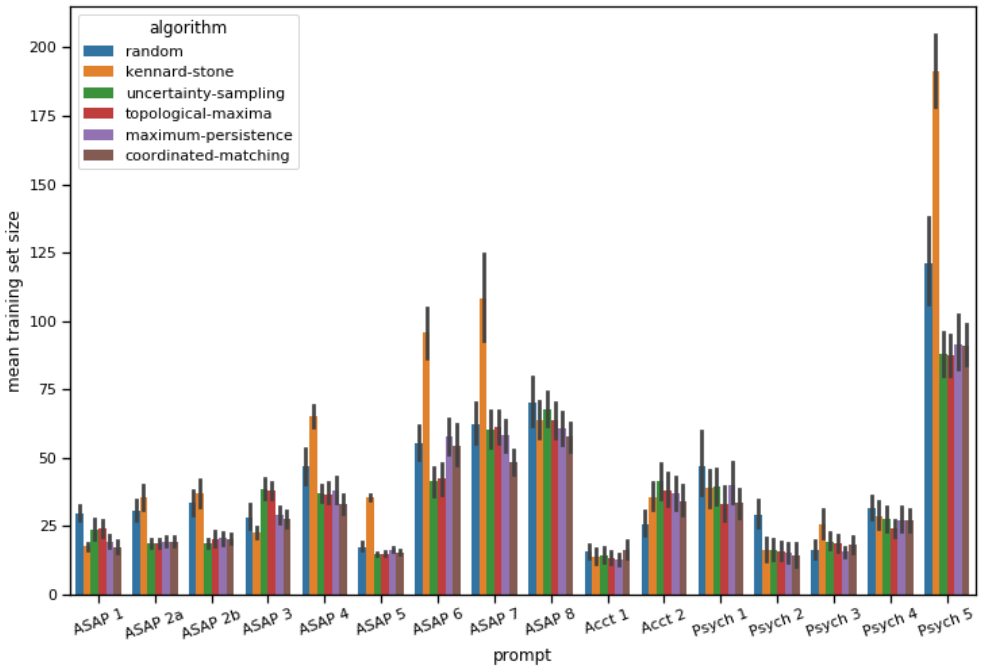

Figure 1. Mean training set size required to reach 95% of full training set performance. Error bars are 95% bootstrap confidence intervals.

Across all prompts and for all active learning algorithms, we can reach 95% of the full training set performance with training sets that are substantially smaller than the full set. For example, on ASAP 4, random selection achieves 95% of the full training set performance after approximately 50 essays have been labeled on average.

On most prompts, the uncertainty sampling approaches perform as well or better than random. The one exception is Accounting 2, where all algorithms perform noticeably worse than random. We also see that, for three prompts (ASAP 5, Accounting 1, Psychology 3), none of our approaches meaningfully outperform random. This may be a boundary effect, as the mean size at which random achieves 95% performance is very small for these prompts.

Kennard-Stone performs notably worse than all other approaches (including random) on ASAP 4, 5, 6, and 7, as well as on Psychology 5. This indicates that Kennard-Stone is poorly suited to our AES task.

We introduced the topological maxima algorithm in an attempt to determine whether or not the persistence aspect of the maximum persistence algorithm was required. From these results, we conclude that it is – topological maxima performs very similarly to uncertainty sampling. This suggests that it is persistence, not just the neighbor graph, that is key to the performance of maximum persistence.

Figure 2 shows, for each prompt, the absolute performance of the active learning algorithms as we increase the number of labeled points. This figure also shows the performance of a model trained on the entire training set. This figure reinforces the conclusion that no algorithms outperform random on ASAP 5, Accounting 1, or Psychology 3 due to a boundary effect – our absolute performance on these prompts starts high and remains high across all training set sizes.

Overall, we conclude that effective AES models can be trained with many fewer scored essays than are used in current systems. Even with random selection, we often only need approximately 50 essays to train a performant model. Determining which active learning approach is appropriate depends on the characteristics we need for our task. For training AES models for formative writing, we want an approach that provides consistently good performance across prompts, even at the expense of missing out on very good performance on some prompts. The very poor performance at low sizes on ASAP 3 for uncertainty sampling and topological maxima makes those approaches less appealing. For our needs, BCM performs very well, in that it provides good performance, relative to random, across all prompts – in AES applications that can tolerate its relatively slow speed, it appears to be a good choice. For applications that require a faster algorithm, maximum persistence also has good overall performance.

Our feature space, model choice, and active learning algorithms were all chosen with the intention of working well for college level formative writing (e.g., the Psychology and

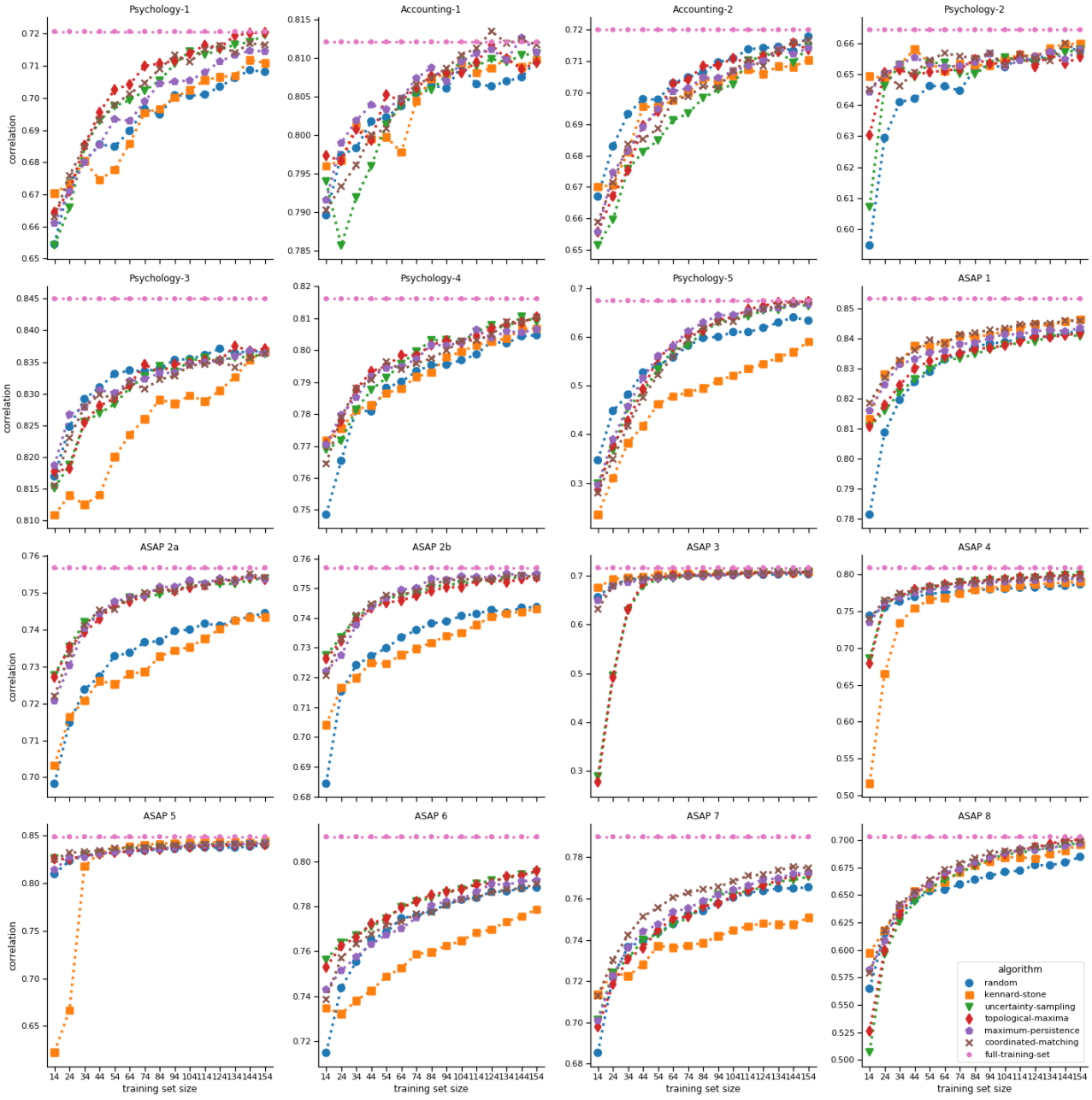

Figure 2. Mean model performance for training set sizes 14 to 154. Error bars omitted for clarity.

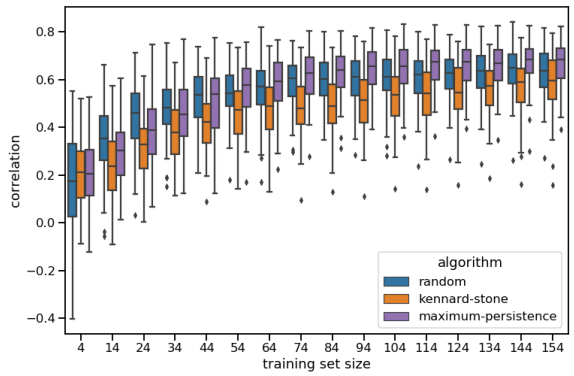

Figure 3. Box plots showing the distribution of model correlations of select active learning approaches across training set sizes for Psychology 5.

Accounting prompts here). That we also achieve strong performance on the ASAP AES dataset suggests that active learning can be useful for AES in general, not just for the specific domains that we are interested in.

Finally, we show the overall distribution of correlations for Psychology 5 across training set sizes for random, Kennard-Stone, and maximum persistence (Figure 3). It is important to be cognizant that there is considerable variability in performance between runs, and that just because a model has been trained on 154 points does not guarantee that it will perform well. A critical aspect of any real-world active learning AES system will be the ability to effectively evaluate model performance.

ACTIVE LEARNING FOR LARGE SCALE FORMATIVE WRITING

Our ultimate goal is to enable instructors to train AES models for their own classrooms. Our approach leverages two existing educational activities: students doing writing and instructors doing scoring. But it optimizes the task by having the instructor do the minimal amount of scoring before the computer takes over the task.

We have developed a software system to enable this vision [3], shown in Figure 4. This figure captures a moment from an instructor interacting with our system during the AES model building process. The instructor is scoring student essays in an order based on a batch provided by active learning, computed after receiving the previous batch of instructor scored essays. We employ the metaphor of training an AI Assistant to help instructors better understand their role in the training process. The current essay, written by an appropriately anonymized student to a neurotransmitters Psychology prompt, occupies the majority of the screen. On the right there is a grid that the instructor uses to assign scores on different rubric traits. At this point in the figure, the instructor has given scores for all traits; for instance, a 3 was awarded for the Organization trait. Across the top of the screen are navigation controls, as well as a timeline indicating the stages in the modeling process

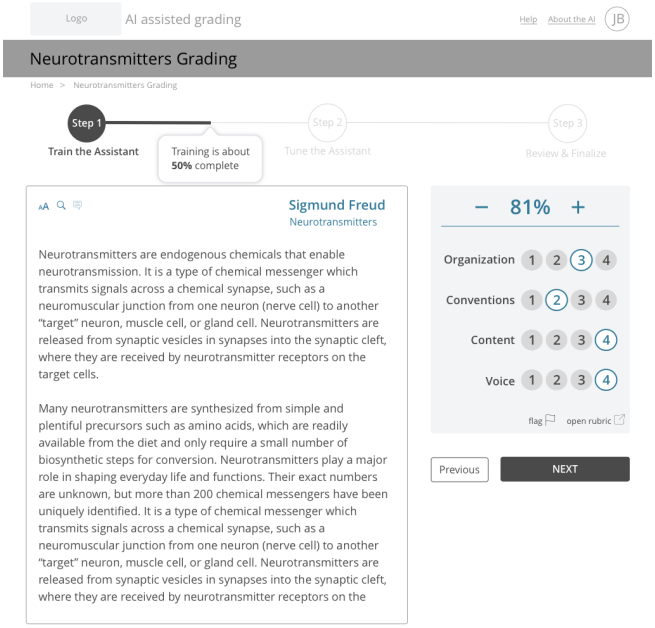

Figure 4. Screenshot of instructor's grading user interface during the "Train the Assistant" phase of modeling. The current student essay determined by active learning is in the box, the grading interface is in the gray shaded area to the right, and navigation and progress timeline are at the top.

and the current progress in the "Train the Assistant" phase. A notification box indicates that, based on a heuristic and current model performance, the instructor has scored approximately 50% of the essays necessary before the model will be ready for the instructor to evaluate.

When the instructor clicks the "next" button, the next selected essay is displayed, and the current essay and scores are queued for the next batch run of modeling and active learning. When that batch runs, the quality of the model is evaluated by cross validation. If it doesn't meet a minimum correlation threshold, the instructor continues scoring, but now with a new batch provided by active learning. If the model does meet the minimum correlation threshold, the instructor moves to the next phase, "Tune the Assistant", in which they are able to evaluate how well their scoring agrees with the AES model. In this phase, the instructor continues scoring essays in batches provided by active learning. But now, after scoring each essay, the instructor compares their scores to the automatically generated scores. Based on this comparison and other feedback available in the interface, such as the overall score distribution, the instructor can evaluate the performance of the AES model. When the instructor determines the model meets their criteria, the remainder of the essays can be automatically scored. In addition, the model is archived so that it can be reused with this prompt in the future.

CONCLUSION

A major barrier to implementing automated essay scoring at scale has been the need to collect large numbers of essays for the training of scoring models. We have shown that batch

mode active learning is an effective technique for reducing the number of essays that must be manually scored in order to train a random forest model that performs well for AES. Such reductions allow AES models to be trained faster and more cheaply, leading to AES models created for and trained by individual instructors. Furthermore, this approach works iteratively with the instructor so that models are built on the fly, allowing the integration of the system into the instructor's existing scoring workflow.

There are many directions for future work that could provide for more robust active learning. We held batch sizes constant in our experiments, but variable batch sizes could be used to both account for current model uncertainty and reduce downtime for the human scorer.

In a classroom setting, there may be multiple scorers available (e.g., multiple teaching assistants in addition to the instructor), which raises a number of issues. To minimize the total number of hand-scored essays, approaches to intelligently partitioning the essays across all scorers could be used. Additionally, some of these scorers may be more or less skilled at scoring for particular prompts, so properly accounting for this distribution of skills would be critical.

Finally, developing good stopping criteria for training, while beyond the scope of this work, is a critical aspect of active learning in real world settings. There is an extensive literature on optimal stopping criteria, but determining how best to apply those techniques in a real-world essay scoring setting is a difficult task. We are currently collecting pilot data with our system in order to help shed light on these questions.

ACKNOWLEDGEMENTS

We would like to thank Mahbub Gani for his assistance in implementing the maximum persistence algorithm.

REFERENCES

1. Yigal Attali and Jill C. Burstein. 2006. Automated essay scoring with e-rater® V. 2. *The Journal of Technology, Learning and Assessment* 4, 3 (2006).
2. Javad Azimi, Alan Fern, Xiaoli Zhang Fern, Glencora Borradaile, and Brent Heeringa. 2012. Batch Active Learning via Coordinated Matching. In *Proceedings of the 29th International Conference on Machine Learning*.
3. Alok Baikadi, Lee Becker, Jill Budden, Peter W. Foltz, Andrew Gorman, Scott Hellman, William Murray, and Mark Rosenstein. 2019. An apprenticeship model for human and AI collaborative essay grading. In *Joint Proceedings of the ACM IUI 2019 Workshops co-located with the 24th ACM Conference on Intelligent User Interfaces (ACM IUI 2019), Los Angeles, USA, March 20, 2019. (CEUR Workshop Proceedings)*, Christoph Trattner, Denis Parra, and Nathalie Riche (Eds.), Vol. 2327. CEUR-WS.org.
4. Stephen P. Balfour. 2013. Assessing Writing in MOOCs: Automated Essay Scoring and Calibrated Peer Review™. *Research & Practice in Assessment* 8 (2013), 40–48.
5. Robert Burbidge, Jem J Rowland, and Ross D King. 2007. Active Learning for Regression Based on Query by Committee. In *Intelligent Data Engineering and Automated Learning - IDEAL 2007*. Springer Berlin Heidelberg, 209–218.
6. Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba. 2009. Analysis of Scalar Fields over Point Cloud Data. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1021–1030.
7. Yuxin Chen and Andreas Krause. 2013. Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization. In *Proceedings of the 30th International Conference on Machine Learning*. 160–168.
8. Abhimanyu Das and David Kempe. 2008. Algorithms for subset selection in linear regression. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. 45–54.
9. Paul B. Diederich. 1966. How to measure growth in writing ability. *The English Journal* 55, 4 (1966), 435–449.
10. Dmitriy Dligach and Martha Palmer. 2011. Good Seed Makes a Good Crop: Accelerating Active Learning Using Language Modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 6–10.
11. Nicholas Dronen, Peter W. Foltz, and Kyle Habermehl. 2015. Effective Sampling for Large-scale Automated Writing Evaluation Systems. In *Proceedings of the Second ACM Conference on Learning @ Scale*. 3–10.
12. Peter W. Foltz, Darrell Laham, and Thomas K Landauer. 1999. Automated essay scoring: Applications to educational technology. *EdMedia: World Conference on Educational Media and Technology* (1999), 939–944.
13. Peter W Foltz, Lynn A Streeter, Karen E Lochbaum, and Thomas K Landauer. 2013. *Implementation and Applications of the Intelligent Essay Assessor*. Routledge Handbooks Online.
14. Steve Graham, Karen Harris, and Michael Hebert. 2011. Informing Writing: The Benefits of Formative Assessment. A Report from Carnegie Corporation of New York. *Carnegie Corporation of New York* (2011).
15. Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006a. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th international conference on World Wide Web*. 633–642.
16. Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. 2006b. Batch mode active learning and its application to medical image classification. In *Proceedings of the Twenty-Third International Conference on Machine Learning*. 417–424.

17. Andrea Horbach and Alexis Palmer. 2016. Investigating Active Learning for Short-Answer Scoring. In *BEA@ NAACL-HLT*. 301–311.
18. R W Kennard and L A Stone. 1969. Computer Aided Design of Experiments. *Technometrics* 11, 1 (1969), 137–148.
19. Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. 2008. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research* 9 (2008), 235–284.
20. Thomas K Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25, 2-3 (1998), 259–284.
21. David D Lewis and William A Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*. Springer-Verlag New York, Inc., New York, NY, USA, 3–12.
22. Ismini Lourentzou, Daniel Gruhl, and Steve Welch. 2018. Exploring the Efficiency of Batch Active Learning for Human-in-the-Loop Relation Extraction. In *Companion Proceedings of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 1131–1138.
23. Chunyong Luo, Yangsheng Ji, Xinyu Dai, and Jiajun Chen. 2012. Active Learning with Transfer Learning. In *Proceedings of ACL 2012 Student Research Workshop (ACL '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 13–18.
24. Dan Maljovec, Bei Wang, John Moeller, and Valerio Pascucci. 2014. *Topology-Based Active Learning*. Technical Report. Scientific Computing and Imaging Institute.
25. Nobal Bikram Niraula and Vasile Rus. 2015. Judging the quality of automatically generated gap-fill question using active learning. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. 196–206.
26. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.
27. Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1160–1170.
28. Y Rubner, C Tomasi, and L J Guibas. 1998. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 59–66.
29. Hinrich Schütze, Christopher D. Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge University Press.
30. Burr Settles. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.
31. Yanyao Shen, Hyokun Yun, Zachary Chase Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep Active Learning for Named Entity Recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017*. 252–256.
32. Mark D. Shermis and Jill C. Burstein (Eds.). 2003. *Automated essay scoring: A cross-disciplinary perspective*. Lawrence Erlbaum Associates, Inc., Mahway, NJ.
33. Mark D. Shermis and Jill C. Burstein (Eds.). 2013. *Handbook of automated essay evaluation: Current applications and new directions*. Routledge, New York.
34. Mark D. Shermis and Brad Hammer. 2012. Contrasting state-of-the-art automated scoring of essays: analysis. In *Paper presented at the National Council on Measurement in Education, Vancouver, BC*.
35. Katrin Tomanek and Udo Hahn. 2009. Reducing Class Imbalance During Active Learning for Named Entity Annotation. In *Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP '09)*. ACM, New York, NY, USA, 105–112.
36. David M. Williamson, Xiaoming Xi, and F. Jay Breyer. 2012. A framework for evaluation and use of automated scoring. *Educational measurement: issues and practice* 31, 1 (2012), 2–13.
37. Edward W. Wolfe. 2004. Identifying rater effects using latent trait models. *Psychology Science* 46 (2004), 35–51.