Trade-offs in Sampling and Search for Early-stage Interactive Text Classification

Zachary Levonian*
University of Minnesota
Minneapolis, Minnesota, USA
levon003@umn.edu

Chia-Jung Lee
Amazon
Seattle, Washington, USA
cjlee@amazon.com

Vanessa Murdock
Amazon
Seattle, Washington, USA
vmurdock@amazon.com

F. Maxwell Harper
Amazon
Seattle, Washington, USA
fmh@amazon.com

ABSTRACT

For many automated classification tasks, collecting labeled data is the key barrier to training a useful supervised model. Interfaces for interactive labeling tighten the loop of labeled data collection and model development, enabling a subject-matter expert to quickly establish the feasibility of a classifier to address a problem of interest. These interactive machine learning (IML) interfaces iteratively sample unlabeled data for annotation, train a new model, and display feedback on the model's estimated performance. Different sampling strategies affect both the rate at which the model improves and the bias of performance estimates. We compare the performance of three sampling strategies in the "early-stage" of label collection, starting from zero labeled data. By simulating a user's interactions with an IML labeling interface, we demonstrate a trade-off between improving a text classifier's performance and computing unbiased estimates of that performance. We show that supplementing early-stage sampling with user-guided text search can effectively "seed" a classifier with positive documents without compromising generalization performance—particularly for imbalanced tasks where positive documents are rare. We argue for the benefits of incorporating search alongside active learning in IML interfaces and identify design trade-offs around the use of non-random sampling strategies.

CCS CONCEPTS

• **Computing methodologies** → **Batch learning**; **Active learning settings**.

KEYWORDS

sampling, classification, interactive machine learning

ACM Reference Format:
Zachary Levonian, Chia-Jung Lee, Vanessa Murdock, and F. Maxwell Harper. 2022. Trade-offs in Sampling and Search for Early-stage Interactive Text Classification. In *27th International Conference on Intelligent User Interfaces (IUI '22), March 22–25, 2022, Helsinki, Finland*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3490099.3511134

*Work conducted during an internship with Amazon.

IUI '22, March 22–25, 2022, Helsinki, Finland

ACM ISBN 978-1-4503-9144-3/22/03.
https://doi.org/10.1145/3490099.3511134

1 INTRODUCTION

Automated classification systems are widely used for reducing human effort and creating structure from diverse data. Supervised machine learning approaches are the go-to approach for automated classification, but collection of labeled data is required in order to train and evaluate the resulting model. Interactive machine learning (IML) systems aim to tighten the loop of labeled data collection and model development in order to quickly establish the feasibility of training a model for a new classification task. Subject-matter experts can use an IML interface to train an initial model and receive immediate feedback on the model's performance. This feedback allows the expert to explore a classification problem before investing in creating large amounts of labeled data, e.g., via crowd-sourcing. In many cases the cost of labeling must be paid before problem feasibility is known, which limits innovation to those technologies that do not require labeled data or for which suitable data already exists. IML systems offer the promise for machine learning novices to apply their subject-matter expertise to classification problems of interest, from the highly individual (personal email categorization) to problems of broad interest (vandalism detection on Wikipedia). An IML data labeling interface can enable experts who understand end-users' needs—including end-users themselves—to quickly train and evaluate a classification model.

While IML techniques are diverse, the exemplar interface for an IML labeling system iteratively samples data for a user to label, trains a model from the provided labels, and displays an estimate of the model's performance so far. Compared with static annotation efforts, IML uses interactivity to promote understanding of the data and to learn an effective classifier with fewer total labels provided [21]. In this paper, we focus on three aspects of interactive labeling systems for text classification: using active learning to select the data that are shown to the user [2], visualizing model performance [61], and incorporating user-guided sampling via full-text search [4]. We investigate these aspects by simulating a user's repeated interactions with an IML system across various sampling and search strategies. We show that these aspects implicate design trade-offs in terms of model performance. See Figure 1 for a demonstration of the trade-offs we investigate in this paper. As a person's

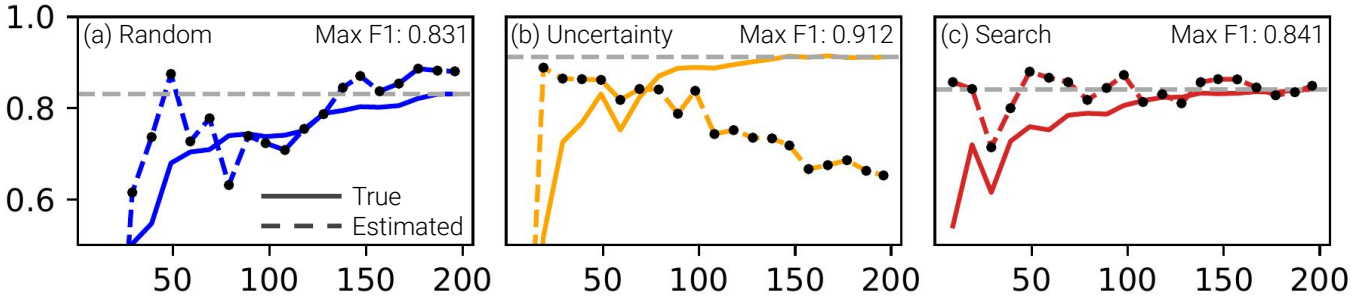

Figure 1: A simulated subject-matter expert uses an IML labeling interface to train a binary text classifier on a novel business problem. In (a), she iteratively labels 10 randomly-sampled documents at a time until she has labeled 200 total. She uses the interface's estimate of the classifier's performance to assess her progress. The performance estimate is computed using cross-validation, so is high variance but is an unbiased estimate of the true generalization performance, which is unknown to the expert but computed in our simulations from held-out data. In (b), she uses the same process, but the IML interface selects the 10 documents for the expert to label at each batch using uncertainty sampling, an active learning technique commonly used in IML labeling interfaces. While the generalization performance of the classifier is higher when active learning is used relative to the performance at the same number of randomly-sampled data, the estimated performance is lower than the true performance and *decreases* as additional data is labeled. With high bias in the performance estimate, the expert cannot accurately assess the feasibility of conducting a larger, more-costly annotation process for the problem. In (c), the expert uses their domain knowledge to search for and label the first 10 documents using a full-text search query before using random sampling, which biases the performance estimate towards initial over-optimism but helps achieve faster model growth and more stable performance estimates. These three examples are the median of 100 simulated runs (by true F1 score at n=200) on the Cat = Books task, introduced in sec. 3.1. Dashed line indicates max F1 at n=200.

initial experience with a labeling interface affects their trust of the system and their assessment of the classification task, we focus on *early-stage* IML: the first minutes or hours of a person's experience with the system, where dozens rather than thousands of labels have been provided. We address three research questions:

RQ1: Which sampling strategies produce models with low generalization error during early-stage IML? The first step in an IML system is the selection of a batch of one or more documents to be labeled. Active learning methods select the data that will most improve the classifier by some metric and are the "cornerstone" of IML systems because they reduce costly human time spent labeling data [18]. However, by intentionally biasing the labeled sample, the resulting classifier may be difficult to evaluate and vulnerable to spurious correlations [37, 45, 72]. We contrast random sampling of unlabeled data with two representative active learning techniques in order to explore the efficacy of these approaches during early-stage learning.

RQ2: What is the impact of non-*i.i.d.* sampling strategies on estimating model performance? Visualizing the model's performance helps the user predict the classifier's likelihood of acceptability for their task [61], helps the user feel the model is improving with their efforts [61], affects the user's trust in the trained model [71], and enables stopping when a target performance is reached [37]. Visualizing model performance requires low-bias *estimates* of that performance, but statistical methods such as cross-validation assume that labeled data is *independent and identically distributed (i.i.d.)*, an assumption that is violated when non-random sampling or user-guided search are used [62]. Further, these methods can produce high-variance estimates in small samples [11]. We compare the bias and variance of cross-validation estimates for the three sampling strategies.

RQ3: Under what conditions can user-guided search be incorporated into early-stage labeling? Guided learning is an alternative to active learning in which the user directly selects data for labeling [4]. Search enables users to apply their domain knowledge to find documents, which is particularly appealing in highly class-imbalanced settings. While guided learning has been successful for several tasks, these approaches have mostly relied on external [36, 41] or visual search tools [31]. Incorporating text search into IML systems may improve classifier performance [60], but the bias introduced by both search and active learning may compound and result in a negative feedback loop wherein poor initial classifier performance leads to ineffective active learning [5]. Previous IML interfaces have used text search to seed the classifier with minority-class documents [60, 64, 67] or to correct for a heavy class imbalance [17]. However, the impact of using these approaches on classifier generalization performance in early-stage labeling is unknown. We use human-generated search queries to evaluate the utility of search across a variety of imbalanced class distributions.

Our simulations with four binary text classification tasks demonstrate a trade-off between improving a classifier's performance and computing unbiased estimates of that performance. We show that supplementing early-stage sampling with user-guided text search *can* effectively "seed" a classifier with positive documents without compromising generalization performance, particularly for tasks with a class imbalance. Our empirical results provide evidence that search can be used alongside active learning in IML labeling interfaces to increase generalization performance. In total, our simulations demonstrate potential costs and benefits to using text search

and active learning for text classification across a wide range of conditions and serve as a necessary precursor to a future user study. We propose guidelines for people creating text classification models and designing IML text classification interfaces.

2 RELATED WORK

Interactive machine learning (IML) [24] is an interaction paradigm describing a broad set of systems where end-users engage with a user interface to iteratively train a machine learning model. Examples of IML systems in everyday use include junk email filtering systems (e.g., "mark as spam") and online recommender systems (e.g., star ratings) [21]. Beyond these common cases, more targeted IML systems have been developed for a broad array of interesting uses, including automatic citation screening for clinical researchers [65], visual recognition of music conductors' movements [57], and recognition of symbolic gestures for user input [46]. IML systems support a variety of machine learning tasks [47], including clustering [15], classification [33], and natural language processing [40]. Dudley and Kristensson provide a survey of research on the design of IML systems [21]. We focus here on IML systems that collect data *labels* from users, as distinct from collecting keywords, explanations, contrast sets, or labeling functions [18, 47].

For IML labeling systems, selecting the sample of data to be labeled is a key concern. Many IML systems incorporate active learning, which refers to techniques for selecting training examples that will provide the most information to a learner, in order to save the end-user time and effort picking the most effective instances [69] and to reduce total labels needed [58]. However, active learning may not be as effective in the cold-start context before the learner is able to make strong predictions [72], and may struggle to locate instances in problems with highly-skewed class distributions or many pockets of rare subclasses [5]. We examine these claims through experiments with two active learning variants (uncertainty sampling [58] and diversity sampling [72]) across different levels of class skew in a cold-start environment.

An alternative to traditional ML sampling techniques is search, where the end-user is able to manually locate instances to interact with. Under extreme skew, "guiding" the learning with user searches has been shown theoretically to improve performance compared to active learning [4, 8]. Prior empirical work has shown gains from user-guided search, especially when the end-user is experienced with the application domain [31, 64, 67]. Moreover, human-computer interaction studies of IML design have found that end-users want the ability "to search through more and varied samples" [50]. However, guided search can lead to biased samples and unnecessary expense for balanced datasets; researchers have investigated higher-level strategies for IML to switch between these paradigms to optimally train a classifier [41]. We run experiments with text classification tasks to better understand the conditions where user-guided search can complement active learning.

Beyond sampling strategies, a key question in the design of IML is how the system should communicate the current state of its model [2]. Much research around these questions has focused on user-centered evaluation of prototype systems to learn about specific design choices. For instance, Amershi et al. [1] study techniques for visualizing the learner's concepts in an image classification system, analyzing several different ways for selecting representative examples that best demonstrate the concept the model has learned. Sun et al. [61] build a prototype system to explore visualization strategies to enhance the end-users understanding of the underlying dataset and the classifier's decisions. In this work, we assume the use of standard classification metrics to reflect the system's performance (e.g., F1 score), and focus on deepening our understanding of how different sampling strategies (including active learning and full-text search) affect the system's ability to accurately estimate model performance.

Table 1: Binary tasks for the Amazon Review dataset. Tasks vary in their positive class proportion (α) and their difficulty. Maximum test set performance is shown in terms of F1 score, recall, precision, and average precision respectively with 10K training documents.

Task	α	$F1_{10K}$	R_{10K}	P_{10K}	AP_{10K}
Is Verified	78.28%	0.8932	0.9471	0.8451	0.9062
Is 5 Star	61.98%	0.8713	0.8948	0.8491	0.9256
Cat = Books	25.11%	0.9273	0.9072	0.9465	0.9720
Cat = Movie/TV	6.45%	0.7670	0.7008	0.8493	0.8285

3 INTERACTIVE LEARNING ENVIRONMENT

We implemented an IML system that samples batches of unlabeled documents and collects document labels from a human user. In order to compare the impact of various sampling and search strategies on classifier performance, we simulate a user's iterative interactions with that system, including providing labels and filtering the unlabeled documents with full-text searches. For each of four binary classification tasks, described in sec. 3.1, we simulate the user's interactions with the IML system as the sampling strategy varies.

First, the IML system retrieves a batch of unlabeled documents for the user to label. A batch here means a set of documents that are sampled and shown to the user simultaneously for annotation. Second, we simulate this human annotation process by providing the known task-specific labels for the batch to the system. Third, after each batch is labeled, a new classifier is trained using the configuration described in sec. 3.2. Fourth, the new classifier is used to compute both a *true* measure of generalization performance from a held-out test set that would not be available to the IML system's user and an *estimated* measure of that generalization performance from the provided labels that would be available—as described in sec. 3.3. Finally, the system samples a new batch from the unlabeled pool and the iterative labeling process continues until the user chooses to stop; our simulated user stops labeling at 200 documents. The strategies used for sampling each batch from the unlabeled pool are described in sec. 3.4. Before each batch, the user can also choose to filter the unlabeled pool with a full-text search query before the sampling strategy is applied. We simulate these search behaviors with strategies for *when* to search—such as only before the first batch to seed the classifier with positive documents—and *which queries* are used, both described in sec. 3.5. Ultimately, we use the described IML system and simulated human user to conduct a

variety of simulations using different sampling and search strategies, as described in sec. 4. An overview of our system is shown in Figure 2.

3.1 Dataset

We conduct our experiments on a dataset of 150 million Amazon customer reviews.[1] We selected this dataset because the purpose of the texts is focused but the topics are diverse. Using metadata for the reviews, we create four binary classification tasks with differing positive class proportions and that variously capture information about the review writer or the reviewed product. For each task, the trained model uses only the review title and text to predict the metadata value. The tasks are: Is Verified (true if the review writer is a verified purchaser of the product), Is 5 Star (true if the writer assigned the product the maximum review score of 5 stars), and two product category tasks. Product category is a broad categorization of the product; we select two at different positive class proportions: Books and Movie/TV. Additional details are in Appendix D.

We randomly sampled 10,000 training reviews to serve as unlabeled documents eligible for sampling during simulation and 10,000 testing reviews for computing generalization performance. Table 1 shows the four tasks and their positive class proportions (α) on the training documents. Table 1 also shows the performance of models trained for each task with *all* training documents available. These scores represent the maximum feasible performance for each task for the learning configuration described in sec. 3.2 and represent a generalization performance target for any classifier trained via the IML interface [37]. To ensure the strongest possible comparison target, we conducted a hyperparameter search to maximize test-set F1 score considering only two parameters: the proportion of minority-class documents to include in the training data $\in [0, 0.5]$—implemented as undersampling—and a weight correction for the undersampling—implemented as BBSC [43]. Note that tasks vary in their modeling "difficulty".

3.2 Feature Representation & Statistical Learner

We use a fixed configuration for all models: logistic regression with ℓ^2 regularization trained on RoBERTa contextualized word embeddings. RoBERTa [44] is a large language model from which we extract feature representations from the first 512 tokens of each document [68], a pre-processing step that can be cached to reduce training time. Contextualized word embeddings from large language models are particularly intriguing in small-sample contexts like IML due to the their few-shot learning capabilities [12], and others have used word embeddings in IML contexts specifically in pursuit of their richness for few-shot learning [6]. To combine the contextualized word embeddings into a fixed-length document representation, we averaged the embeddings for each token (i.e., mean pooling, see [59]). We did not pre-train RoBERTa on our dataset.

To set the feature representation and statistical learner, we conducted experiments on the Is 5 Stars and Is Verified tasks.[2] For the feature representation, we compared bag-of-words with three RoBERTa-based variants: mean pooling, max pooling, and the representation of the CLS token. For the statistical learner, we compared logistic regression, gradient-boosted decision trees, linear SVMs, and two-layer feedforward neural nets with a sigmoid head. Between 20 and 500 labeled training documents, mean pooling dominated for all learners; bag-of-words may be more effective in the extremely low-sample context ($n < 20$), but was outclassed by all RoBERTa representations as n increased. While the neural net outperformed logistic regression by as much as 0.1 F1 score at some sample sizes, we found this behavior to be inconsistent: performance was highly sensitive to the Adam hyperparameters and the hidden layer size. Thus, we opt for logistic regression, which has the additional benefits of being fast to train and well-calibrated, both beneficial traits in an IML context.[3] It is heartening that the rich RoBERTa representations can be effectively incorporated via a quick-to-train model.

3.3 Model Metrics & Estimation

We consider two distinct modeling goals for an IML interface: (a) maximizing model generalization performance and (b) minimizing the bias and variance of *estimates* of generalization performance.

3.3.1 True performance. We define generalization performance as the performance of the current classifier on the full document pool, where the goal of the user is generally to train a classifier that maximizes generalization performance. In practice, generalization performance is estimated from empirical performance on labeled data. We use performance on the held-out test set of 10,000 documents as a low-bias proxy for "true" generalization performance. This true performance estimate enables us to compare the impact of various sampling and search strategies on generalization performance.

3.3.2 Estimated performance. While training a classifier that maximizes generalization performance may be the user's top priority, users of IML interfaces also desire feedback on their progress and an estimate of the classifier's efficacy and uncertainty on the task [21]. Further, accurate estimates of model performance as more data is collected enables estimating the shape and power-law exponent of the learning curve [32, 62]. This information can be used to predict the amount of additional data that needs to be labeled to reach a target performance [42]. To support these goals, an estimate of model generalization must be computed from the small number of labels provided by the user.

One approach is to require the user of an IML interface to conduct preliminary or intermittent labeling on randomly-sampled data in order to construct a test set for evaluation. For example, Sun et al. task their users with first labeling 100 documents and then visualize model performance on those 100 documents [61]. However, upfront labeling is unappealing for exploratory labeling tasks, as the user is unable to get early feedback on the performance of the model—one of the core motivations for exploratory labeling in the first place. Further, the user's understanding of a task can shift as they explore the data [38], which invalidates any existing labeled test data and motivates identifying concept or task shifts as quickly

[1] https://s3.amazonaws.com/amazon-reviews-pds/readme.html
[2] Details in Appendix B.
[3] We used scikit-learn for model optimization and prediction [53], NumPy [30] for data manipulation, and Matplotlib [35] for producing visualizations.

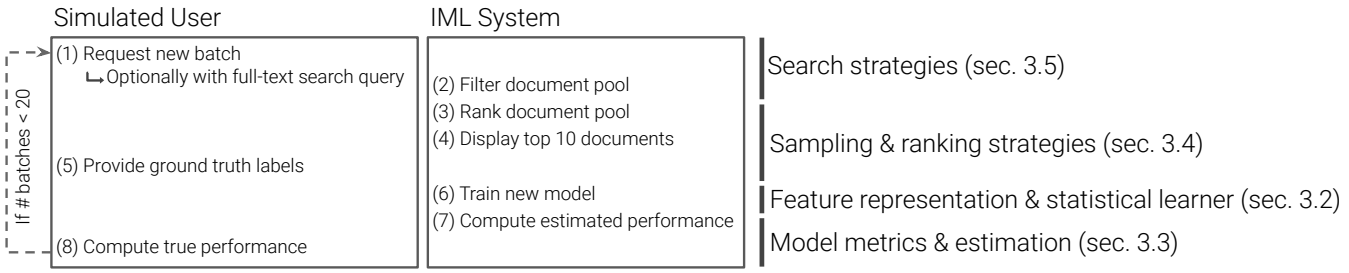

Figure 2: An overview of the system components in our simulations.

as possible. Instead of a separate test set, we estimate generalization performance from the labels provided so far.

Cross-validation (CV) is the primary approach used for estimating generalization performance when a defined validation or test set is not available. k-fold CV randomly splits the data into k equal-size subsamples and treats each subsample in turn as the validation data for a model trained from all other subsamples. As k increases, estimate variance decreases but computational cost increases [20]. We set $k = 50$ in our simulations.[4]

3.3.3 Metrics. While an IML interface might selectively display metrics of interest to the user, we focus on F1 score as a metric that is robust to class imbalance and incorporates both the positive and negative class. Where possible, we report precision and recall values as well, at a 0.5 decision threshold. We report average precision (AP) where appropriate. Equivalent to the area under the Precision-Recall curve, AP provides a measure of model calibration and model performance as the score threshold varies.

3.4 Sampling & Ranking Strategies

Sampling from a pool of unlabeled documents can be viewed as a ranking problem. Traditional random sampling of n documents constitutes selecting the top n documents of a random ranking. Active learning is a machine learning paradigm that ranks a pool of unlabeled documents based on some measure of their expected utility to the learning problem. In contrast, information retrieval methods for full-text search rank documents based on the expected relevance of that document to the searcher. In IML interfaces that support text search, these two approaches are potentially in tension. Our IML system ranks all unlabeled training documents using one of the methods described below and selects the top n as a batch to receive labels.[5] Active learning is a large field, so we include two baseline approaches that reflect two core pillars of active learning.[6] The sampling approaches we consider are:

(1) Random. To construct a batch, random sampling selects n documents uniformly at random from the unlabeled training pool—equivalent to ranking the documents by a random score.

(2) Uncertainty. Uncertainty sampling is a widely-used active learning technique that ranks documents according to their "uncertainty" to an existing classifier [58]. In the binary case, documents are scored using their normalized predictions $f(d)$ according to $1 - |f(d) - 0.5|$. For logistic regression, the highest scores are those closest to the decision boundary. Uncertainty methods can do badly in the first few iterations of batch-mode active learning [72]; we can get caught in a "vicious cycle" where uninformative labels lead to an uninformative model and thus further bad ranking [5, 67]. If a classifier is not available—i.e., when fewer than two positive and two negative documents are labeled—random sampling is used instead.

(3) Diversity. Diversity sampling attempts to create batches in which the selected documents are dissimilar to each other [23]. Diversity-based methods generally involve computing the distance between documents in their feature representations [72]. We use a baseline evaluated by Yuan et al. as similarly-effective to more complex approaches [72]: run k-means for 10 iterations on the unlabeled training pool's RoBERTa embeddings, then rank documents by their distance to the nearest cluster center and select the documents closest to each center to fill the batch. While Yuan et al. set k equal to the batch size $|B|$ and returned the nearest neighbor of all cluster centers, their batch size of 100 is much larger than most IML interfaces. We observed significant performance variation for small batch sizes, so we instead allow $k > |B|$ and fill each batch by selecting randomly from the k documents closest to the cluster centers. Once k documents are sampled after $k/|B|$ batches, we re-execute k-means on the remaining unlabeled pool and repeat. We evaluated $k \in [10, 200]$, finding minimal differences above 50 and opting for $k = 100$ for consistency with Yuan et al. In IML interfaces, k-means-based diversity sampling has been previously used by Park et al. to identify texts representative of a dataset [52].

(4) BM25. When the user conducts a search via a text query, ranking strategies that attempt to model the relevance of each document to the searcher can be used. In contrast, the three prior ranking approaches model the "relevance" of each document to the *model*, which leads to concerns that using human relevance ranking methods may degrade sample efficiency and thus model performance. We chose BM25 as a strong, widely-used relevance baseline [3, 55]. To score documents using BM25, we use the Anserini system with default parameters for BM25 to retrieve and score documents given

[4] If the number of labeled data n is < 50, we set $k = n$ instead. i.e., $k = \min(50, n)$

[5] While prior work has found active learning gains using batch-aware strategies beyond top-n selection, these gains are generally modest [14]. Thus, we use top-n rankings as the more parsimonious and widely-used strategy.

[6] Some active learning approaches use importance sampling to explicitly correct for the sampling bias introduced [7]. Such approaches give theoretically-unbiased estimates of the risk [26]. We implemented LCB-AL [27], the current state-of-the-art among pool-based active learning algorithms that provide these unbiased estimates. Empirically, in the early-sample setting we found LCB-AL to perform equivalently to random sampling in terms of true performance and worse in terms of estimated performance. (See Appendix F.) For simplicity of presentation, we omit LCB-AL from our primary results.

Table 2: Top human queries for each task, sorted by positive class proportion in the filtered pool

Task	Human Queries
Is Verified	confirmed purchase, verified purchase, Returning the product, i purchased this several times
Is 5 Star	Excellent, best, Great, recommend, highly recommended, happy, excellent quality, amazing product
Cat = Books	Author, novel, book, fiction, page turner, reading, detective stories, an engaging story, non-fiction
Cat = Movie/TV	IMDB, Oscar, Actor, actress, films, movies, Movie, hilarious episodes, watching, re-watch, TV shows

a text query [70]. It is unclear if search users in an IML context perceive relevance-based ranking to be important or desirable [56], but use of such rankings may be important for meeting users' expectations around search interfaces. By using BM25, we quantify the extent to which relevance-based ranking affects model generalization performance in the early-stage IML context.

3.5 Search Strategies

We evaluate the use of guided learning for early-stage IML through full-text search over the training documents. In this environment, a search query represents a filtering of the unlabeled training document pool to a subset on which the sampling strategy is subsequently applied. Thus, we explore two considerations for modeling a user's search process: *when* to search and *which queries* to use. A summary is shown in Figure 3. The default behavior is to conduct no search, such that the sampling strategy ranks every unlabeled document. Under what conditions should a search be conducted? We consider the following conditions:

(1) To seed the classifier. (Seed Search.) Search until two positive documents are labeled, which is the threshold for training an initial classifier. For tasks with a high class imbalance, random or diversity sampling may retrieve a low proportion of positive documents. Uncertainty sampling cannot function at all until a classifier is trained. Thus, we use search to address this cold-start problem and quickly train an initial classifier [4]. This search strategy was previously used in [64].

(2) To correct a class imbalance. (Imbalance Search.) Search when fewer than 25% of labeled documents are positive, with the intent to identify additional positive documents. Extreme class imbalance can make learning an effective model challenging [43], so addressing a class imbalance via search has the potential to satisfy both the user and the needs of the statistical learner. This search strategy is encouraged by Chew et al.'s IML interface to "fix skew" [17].

(3) At random. (Random Search.) Users of IML systems with search features have diverse motivations for searching beyond maximizing the benefit of new labels to the model. To represent other motivations that might arise in actual practice, we choose to search (or not) at each batch, with search probability 20%.

In practice, humans select queries in diverse ways. In an IML context, the user's familiarity with the task and the dataset may affect their ability to construct queries that retrieve positive-class documents that can improve the classifier. To reflect that variance, we conducted experiments with both human-solicited queries and generated queries. We consider two primary[7] query sets:

(1) Human. We surveyed five working data scientists familiar with the dataset to provide 3-10 text queries that they would use to look for documents in the positive class for each of the four tasks (see Appendix C). The resulting set included between 15 and 25 queries for each task (samples shown in Table 2). No filtering was done on the queries; two queries returned no positive documents in their respective task.[8] To conduct searches during our simulations, we select a query at random in each batch for which the search condition is met. The remaining unlabeled training pool is filtered to contain only the documents retrieved by Anserini given that query (i.e., documents with a positive BM25 score). Multi-token queries are handled via the Anserini default [70]. Across all tasks, the median human query retrieves 606 training documents and increases the positive class proportion in the filtered pool by +10%, which indicates that human queries are generally effective at identifying additional minority-class documents. If a search would retrieve fewer than 5 documents (because those documents have already been labeled in a prior batch), a different query is selected at random from the set.

(2) Oracle. What search results are "optimal" from a modeling perspective? Balanced samples with an equal number of positive and negative training points are easiest for modern statistical learners to produce highly-discriminative classifiers [43]. Thus, we include an "oracle" search that returns a stratified random sample with an equal number of positive and negative documents. Notably, our oracle search reproduces the guided learning approach used by Attenberg and Provost in their original simulations [4]. Our study extends beyond this idealized representation to compare against specific search queries.

4 METHODS

We conducted simulations on the Amazon Reviews dataset to explore the varying impact of sampling and search strategies on true and estimated performance. We report results for 20 batches with a batch size of 10, for a final classifier trained with 200 labels. Similar to Englhardt et al., we found that increasing batch size up to about 50 had a minimal impact on performance [23], suggesting that batch size can be fixed according to design concerns without negatively impacting the classifier. For all reported simulations, we computed 100 independent runs. Within each run, we pool the predictions from all CV folds after each batch to compute the performance metrics [25]; then, for each metric, we compute the mean and 95% CI for each batch across all runs.

To address RQ1, we ran simulations using the random, uncertainty, and diversity sampling strategies, computing true generalization performance on the held-out test data after each batch. To

[7] See Appendix E for additional query sets.

[8] "Rotten tomatoes" and "cinema" (Cat = Movie/TV).

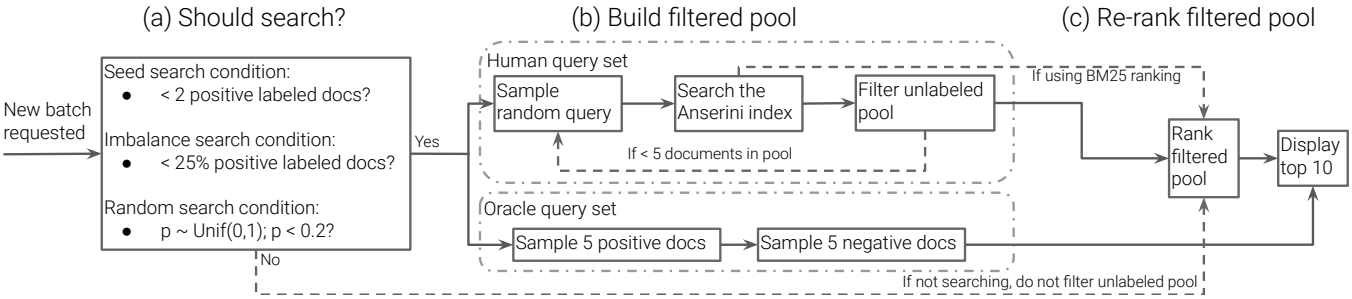

Figure 3: Method used to simulate full-text search.

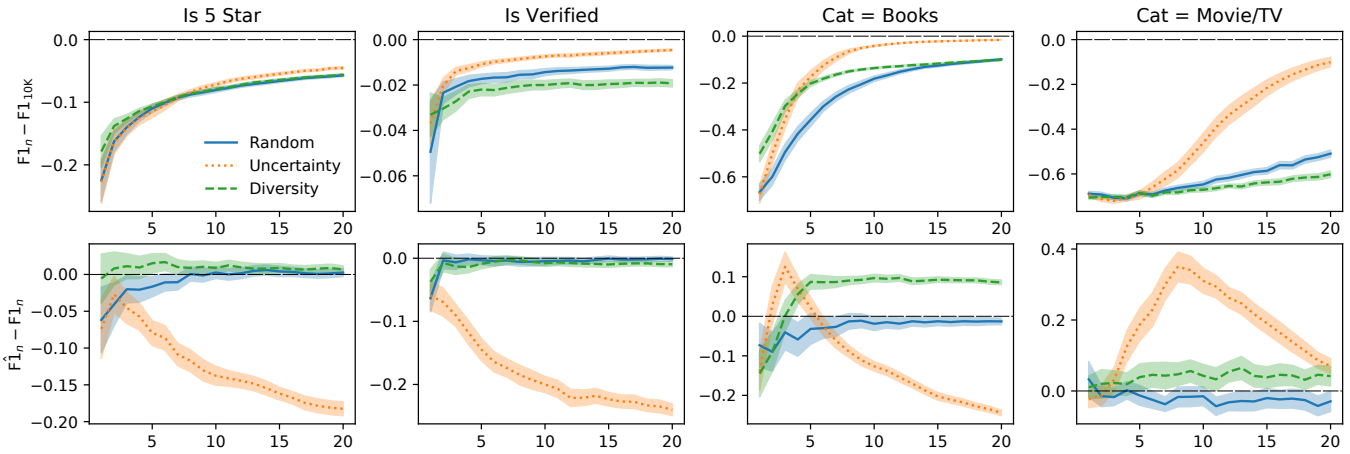

Figure 4: True and estimated performance for three tasks and sampling strategies. True performance (top row) is shown as the difference between the test performance on the model trained with all training data ($F1_{10K}$ in Table 1) and the test performance after n labeled data are collected ($F1_n$) over $\frac{n}{10}$ batches (X axis). The IML system is striving for a true performance difference of 0 (dashed line) in as few batches as possible. Estimation performance (bottom row) is computed as the difference between the cross-validation estimate that uses only the labeled data ($\hat{F1}_n$) and the performance on the held-out test data ($F1_n$). A difference of 0 means no estimation bias. Shading denotes 95% CI for the mean difference in F1 scores over 100 runs.

address RQ2, we computed estimates of generalization performance using cross-validation after each batch. We compute *estimation bias* as the difference between the estimated and the true performance. We compute the variance for the estimation bias using the 100 independent random runs, an approach used by Hanczar et al. [29]. To address RQ3, we repeated experiments from RQ1 and RQ2 with the addition of search. We primarily focus on seed search: searching in the first batches to identify positive labels quickly so that a model can be trained.

5 RESULTS

5.1 RQ1: Sampling strategies

Table 3 shows true model performance after 10 and 20 batches. Note the low performance for the task with the highest class imbalance, Cat = Movie/TV; we will focus on addressing this imbalance in RQ3, using search to address the cold-start problem. The top row of Figure 4 plots the true performance over all 20 batches. We note that Is Verified is an "easier" learning problem; even classifiers with 10-20 labeled points achieve high performance relative to the classifier trained with all 10,000 training points. In the other "harder" problems, we note that uncertainty sampling produces classifiers with the highest true performance.

5.2 RQ2: Estimation bias and variance

While uncertainty sampling produced classifiers with the best true performance relative to random and diversity sampling, how well can that performance be estimated from the labeled data? The second row of Figure 4 plots estimation bias, revealing that random sampling achieves the least biased performance estimation of the three methods. We define a pessimistic approach as one that consistently underestimates true performance, while an optimistic approach consistently overestimates. After 20 batches, uncertainty sampling produces a pessimistic F1 estimate more than 0.1 lower than the true performance across the first three tasks.[9] Additionally,

[9]Despite the pessimistic trend observed with uncertainty sampling, estimated performance is optimistic in the first five batches for Cat = Books and in all of the batches for Cat = Movie/TV. One explanation suggests that optimism and pessimism occur when data are selected far from or close to the optimal decision boundary respectively [37].

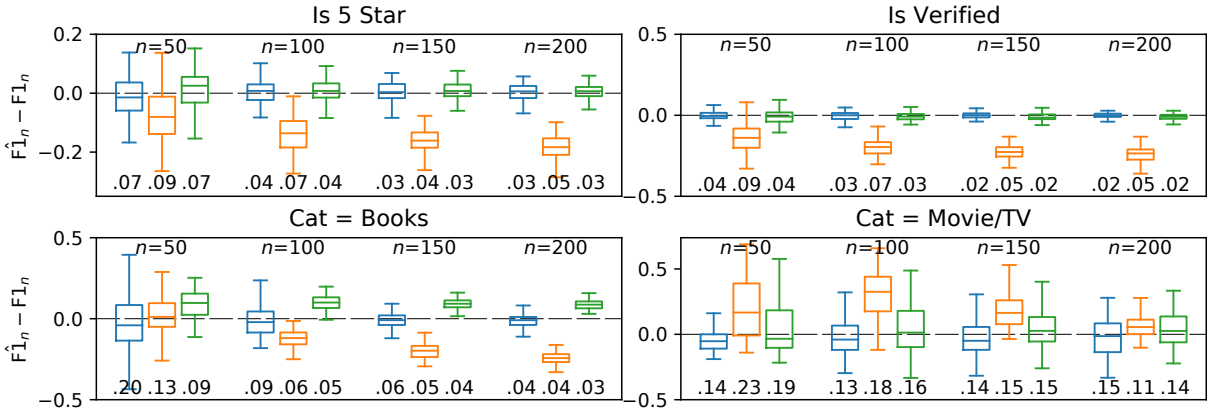

Figure 5: Bias and variance of cross-validation estimates of $F1_n$ for two tasks. Box and whiskers are computed after 5, 10, 15, and 20 batches over 100 runs. In each task/batch triplet, the three boxes from left to right are random, uncertainty, and diversity sampling. Sample standard deviations are shown below each box.

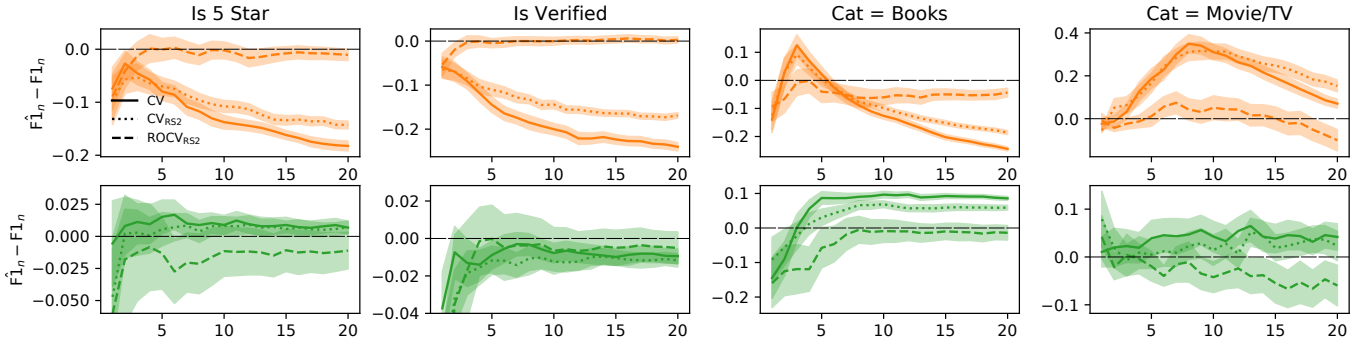

Figure 6: Bias of cross-validation estimated performance as random sampling is introduced. CV gives the default strategy shown in Figure 4. RS2 indicates that 2 of the 10 documents in each batch are sampled at random rather than with uncertainty or diversity sampling. ROCV ("random-only" CV) uses only predictions on the randomly-sampled documents to estimate performance.

diversity sampling produces optimistic estimates for Is 5 Star and Cat = Books.

In addition to low bias, we also want performance estimates to have low variance. Figure 5 shows the bias and variance together for each sampling strategy. While the variance of all estimators shrink as additional labeled data is acquired, across all tasks and sampling strategies the estimated performance has a standard deviation >0.2. The estimate variance for uncertainty sampling is consistently larger than the others. For Cat = Books, the diversity sampling variance is lower than random sampling, which suggests a bias/variance trade-off depending on the sampling strategy and task.[10]

How to address the high estimation bias introduced by non-random sampling strategies? One strategy to decrease bias while still gaining the generalization benefits observed from uncertainty sampling is to increase the proportion of data that is randomly sampled. Figure 6 shows a variant of cross-validation that samples 20% of each batch randomly and then estimates performance only on those randomly-sampled documents. This approach reduces the bias of uncertainty sampling, but at the cost of increased variance due to the smaller sample size: another estimation trade-off.

5.3 RQ3: Search

Does search improve generalization performance? We conducted simulations with the search strategies and query sets defined in sec. 3.5. Table 4 compares true model performance for seed search vs the no-search defaults presented in Table 3. We find that seed search has minimal impact for the first three tasks, but substantially improves true performance for Cat = Movie/TV—the task with the lowest positive class proportion. For Cat = Movie/TV, while the greatest absolute increases in true performance occur with diversity and random sampling only the uncertainty sampling approach produces a reasonable F1 score. We will omit further discussion of diversity

Thus, the early optimism for Cat = Books may be due to weak initial classifiers that misrank uninteresting data that are far from the eventual decision boundary [5].

[10]We observe a similar trade-off with alternative estimation strategies. In addition to cross-validation, we conducted experiments with two bootstrapping strategies—out-of-bag and .632 [22, 29]—finding that they reduced estimate variance at the cost of increased bias and computation time.

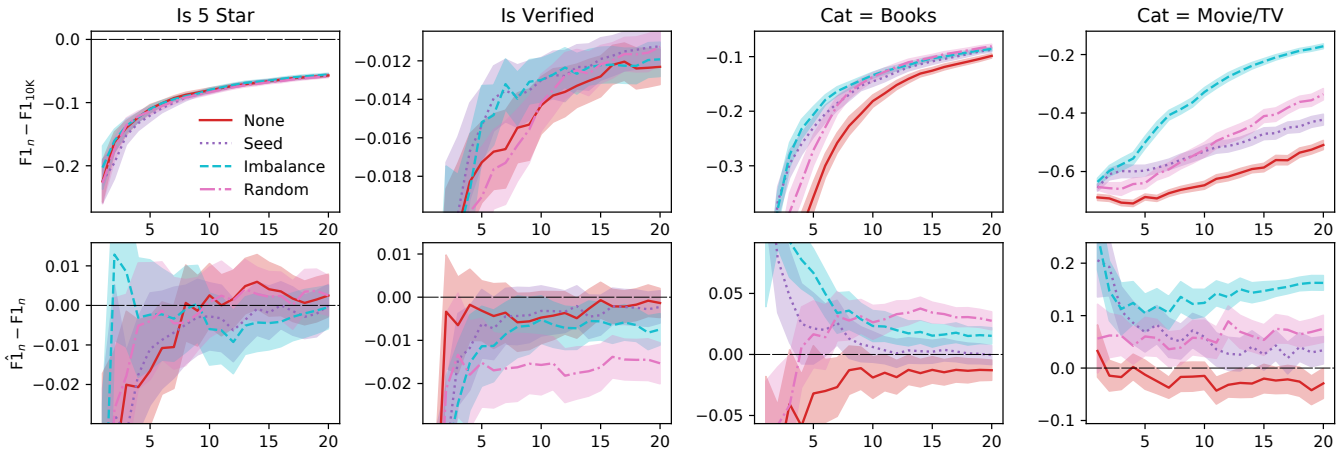

Figure 7: Search + random sampling: true performance ($F1_n - F1_{10K}$) and estimation bias ($\hat{F1}_n - F1_n$) for search and random sampling. Each line captures a different search strategy. The no-search baseline is from Figure 4.

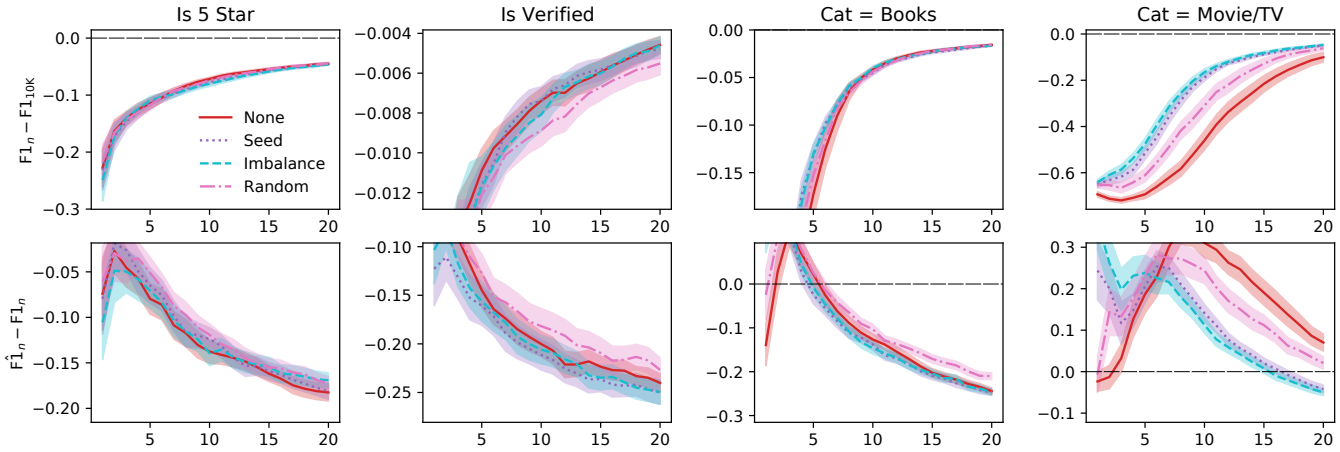

Figure 8: Search + uncertainty sampling: True performance ($F1_n - F1_{10K}$) and estimation bias ($\hat{F1}_n - F1_n$) for search and uncertainty sampling. Each line captures a different search strategy. The no-search baseline is from Figure 4.

sampling in the rest of this section, as it performs qualitatively similar to random sampling while introducing estimation bias. Does it matter how the filtered pool is ranked? Table 6 compares BM25 ranking to random ranking in terms of true F1 after 20 batches, demonstrating that the use of BM25 ranking does not damage true performance—with the exception of Cat = Movie/TV when random sampling is used. This result suggests that using human relevance proxies to rank a search-filtered pool of documents in an IML interface is unlikely to damage generalization performance.

Comparing the human query set to the oracle queries, the two perform similarly. In fact, oracle seed search produces lower performance gain than human seed search for Cat = Movie/TV when random and diversity sampling are used (Table 4). This result suggests that any benefit from a randomly-filtered pool is offset by inducing label shift between the train and test distributions and implies that Attenberg and Provost's results may have been pessimistic relative to actual human-solicited searches [4]. As results are qualitatively similar between the two query sets, we focus on human queries only for the rest of this section.

Compared to seed search, imbalance and random search perform similarly for the first three tasks. The top row of Figures 7 and 8 show true performance in the same style as Figure 4 for random and uncertainty sampling respectively. Table 5 shows the relative differences between the search strategies after 20 batches. For Cat = Movie/TV, "more" search is generally better: with random sampling, imbalance search results in the largest number of searches and so produces the greatest boost in F1 score. With uncertainty sampling, early search (from the seed and imbalanced search strategies) helps uncertainty sampling scale faster than not searching.

While search shows minimal impact on true performance—except for Cat = Movie/TV—search does induce additional estimation bias. The second row of Figures 7 and 8 reveal a pattern of higher estimation bias and variance. In the two category tasks, use of search induces a significant optimism bias in early batches. For seed search,

Table 3: True model performance without search.

			F1	R	P	AP
Is 5 Star	n=100	Random	0.7908	0.8431	0.7490	0.8409
		Uncertainty	**0.7995**	0.8451	0.7621	0.8353
		Diversity	0.7927	0.9024	0.7081	0.8399
	n=200	Random	0.8143	0.8524	0.7811	0.8685
		Uncertainty	**0.8261**	0.8609	0.7948	0.8715
		Diversity	0.8152	0.8981	0.7472	0.8689
	n=10K	Random	0.8713	0.8948	0.8491	0.9256
Is Verified	n=100	Random	0.8789	0.9343	0.8303	0.8857
		Uncertainty	**0.8858**	0.9511	0.8293	0.8854
		Diversity	0.8732	0.9149	0.8361	0.8889
	n=200	Random	0.8809	0.9356	0.8326	0.8908
		Uncertainty	**0.8886**	0.9493	0.8354	0.8932
		Diversity	0.8739	0.9087	0.8423	0.8924
	n=10K	Random	0.8932	0.9471	0.8451	0.9062
Cat = Books	n=100	Random	0.7458	0.6521	0.8860	0.8732
		Uncertainty	**0.8857**	0.8581	0.9160	0.9409
		Diversity	0.7911	0.7495	0.8405	0.8703
	n=200	Random	0.8287	0.7650	0.9065	0.9188
		Uncertainty	**0.9115**	0.8845	0.9404	0.9567
		Diversity	0.8261	0.7915	0.8652	0.9037
	n=10K	Random	0.9273	0.9072	0.9465	0.9720
Cat = Movie/TV	n=100	Random	0.1201	0.0975	0.2955	0.1960
		Uncertainty	**0.3067**	0.2271	0.7077	0.4398
		Diversity	0.0967	0.0850	0.2136	0.1514
	n=200	Random	0.2578	0.2043	0.4203	0.3074
		Uncertainty	**0.6666**	0.5647	0.8394	0.7216
		Diversity	0.1656	0.1289	0.2924	0.2238
	n=10K	Random	0.7670	0.7008	0.8493	0.8285

this bias decreases as additional data is labeled via random sampling. But if the user continues to search, as captured by the random search strategy, unknown estimation bias may be incurred. If uncertainty sampling is used, the estimation bias added by use of search is small relative to the bias incurred by use of uncertainty sampling; thus, if estimation bias is not a priority, use of search is unlikely to harm true performance.

6 DISCUSSION

The consistent take-away of our simulation results is a trade-off between improving a classifier's performance and computing unbiased estimates of that performance. Active learning and search can improve generalization performance but will introduce estimation bias. In the early-stage learning context, uncertainty sampling consistently dominated random and diversity sampling approaches at the cost of a high performance estimation bias. The consistent pessimistic trend induced by uncertainty sampling may be particularly discouraging to a user and lead them to prematurely conclude that the task is infeasible. We show that sampling some data randomly can decrease this bias (Fig. 6) while still maintaining the generalization benefits of uncertainty sampling. Future work could further explore statistical methods that use knowledge of the sampling method to correct for this bias [7, 37], so that IML system designers and users are not forced to trade off between the performance benefits of active learning and low estimation bias.

Our results show that full-text search can effectively "seed" a classifier with positive documents without compromising generalization performance—particularly for class-imbalanced tasks. However, as with active learning, we observe a trade-off between generalization performance and estimation bias. IML systems for text classification might balance a need for strong generalization and estimation by limiting usage of search or non-*i.i.d.* sampling. For example, Lin et al. proposed a learned bandit model to choose *for* the user whether to search and how to sample at each batch [41], while Wall et al. allowed the user to choose among sampling and search options but also provided pop-up system guidance to help users choose [64]. How to design IML interfaces to educate users on the trade-offs involved in these choices is an important direction for future work. In IML contexts where unbiased estimates of classifier performance are *not* important, we show that search can help improve performance on tasks with a low positive class proportion. In those contexts, search can support both the finding of positive class examples and exploration of the underlying data [50]. Search is also used for non-text data (e.g., [13, 31]), and we expect our search results to generalize beyond text classification. When classifier performance estimates *are* important, random sampling after a search can reduce any introduced bias (Fig. 7) or performance estimates can be computed using only non-searched labels (to decrease bias at the expense of additional variance).

6.1 Guidelines

We summarize our findings as guidelines for both *modelers*—people creating text classification models—and *designers*—people designing IML text classification interfaces.

6.1.1 Guidelines for modelers.

(M1) If annotated model performance estimation is important, minimize non-random sampling during annotation. If performance estimation can be deferred until later in the process, evaluation data can be annotated later or annotated jointly with the training sample [48].

(M2) If using uncertainty sampling, expect pessimistic estimates of model performance. Annotating a randomly-sampled evaluation set first—as Sun et al. do [61]—avoids inaccurate estimates during active learning.

(M3) If the expected positive class proportion is low, use search to establish an initial set of positive examples. Positive samples enable a model to be trained, necessary both for model-based active learning methods to be used and to produce any performance estimate [17].

(M4) If search is used during annotation, expect increased variance in the model's performance estimate. While search is desirable in IML contexts [50], use of search may produce unpredictable performance estimates; annotating a larger random sample will increase the stability of the estimate.

Table 4: Does the use of search for model seeding improve the resulting classifier? $F1_{n=200}$ for each task and sampling strategy when seed search is not used, and the change in performance (Δ_{Search}) when search is used in the first batch to seed the classifier. Differences > 0.01 are bolded.

Task	Query Set	Random		Uncertainty		Diversity	
		$F1_n$	Δ_{Search}	$F1_n$	Δ_{Search}	$F1_n$	Δ_{Search}
Is Verified	Oracle	0.881	-0.001	0.889	-0.001	0.874	-0.004
	Human	0.881	+0.001	0.889	-0.000	0.874	-0.000
Is 5 Star	Oracle	0.814	-0.001	0.826	-0.001	0.815	+0.001
	Human	0.814	+0.000	0.826	-0.000	0.815	-0.000
Cat = Books	Oracle	0.829	+0.006	0.912	-0.000	0.826	+0.004
	Human	0.829	**+0.011**	0.912	-0.001	0.826	+0.006
Cat = Movie/TV	Oracle	0.258	**+0.067**	0.667	**+0.051**	0.166	**+0.078**
	Human	0.258	**+0.087**	0.667	**+0.051**	0.166	**+0.116**

Table 5: How does seed search compare to imbalance and random search? $F1_{n=200}$ for each task when human seed search is used, and the change in performance when imbalance ($\Delta_{Imbalance}$) or random (Δ_{Random}) search are used instead. Differences > 0.01 are bolded.

Task	Random			Uncertainty		
	$F1_n$	$\Delta_{Imbalance}$	Δ_{Random}	$F1_n$	$\Delta_{Imbalance}$	Δ_{Random}
Is Verified	0.882	-0.001	-0.000	0.888	+0.000	-0.001
Is 5 Star	0.815	+0.001	-0.001	0.826	-0.001	+0.001
Cat = Books	0.840	+0.002	+0.006	0.911	+0.000	+0.001
Cat = Movie/TV	0.344	**+0.251**	**+0.087**	0.717	+0.002	**-0.012**

Table 6: Does the use of BM25 for ranking seed search results damage the true performance of the resulting classifier? $F1_{n=200}$ for each task when human seed search is used, and the change in performance (Δ_{BM25}) when BM25 is used to rank the filtered pool rather than a random ordering. Differences > 0.01 are bolded.

Task	Random		Uncertainty	
	$F1_n$	Δ_{BM25}	$F1_n$	Δ_{BM25}
Is Verified	0.882	-0.001	0.888	-0.000
Is 5 Star	0.815	-0.002	0.826	-0.000
Cat = Books	0.840	-0.002	0.911	+0.001
Cat = Movie/TV	0.344	**-0.013**	0.717	+0.001

6.1.2 Guidelines for IML text classification interface designers.

(D1) Don't show the user a live estimate of model performance if it is likely to be inaccurate. Users make decisions based on performance estimates, including when to stop annotating, that may be compromised by inaccurate estimates [21].

(D2) Discourage search and non-random sampling if model performance estimates are important to the user. Interface suggestions that describe the impact of these suggestions can help users make informed choices [64].

(D3) Encourage search for positive samples early in the annotation process to increase generalization performance. We observed performance benefits to early searching, which Ng et al. further suggest to be desirable as a component of a user's iterative sensemaking process for a new classification task [50].

(D4) Encourage use of non-random sampling when true performance is the primary or only objective. In general, active learning and particularly uncertainty sampling will increase true performance.

6.2 Societal Impacts

Interactive machine learning makes it easier for non-experts to build and evaluate automated classification systems. IML labeling interfaces contribute to a trend of efforts to democratize AI. Making it easier for individuals to provide input to machine learning systems—e.g., by providing labels—can enable a culture of participatory machine learning [28]. Unfortunately, it is not clear if this trend will have positive effects on society. If IML systems enable people to create more classifiers, it enables those people to automate processes that promote and reflect their personal values but are harmful to society as a whole. For example, a single Wikipedia editor can challenge the consensus of other editors one edit at a time, but easy access to an IML interface for training a classifier could enable them to identify and disrupt thousands of pages. Even where classification is appropriate and beneficial, harms may result if adaptive, contestable human processes are replaced by rigid automated classifiers. Considering our specific approach to IML, we use RoBERTa embeddings as input features, which are known to capture various stereotypical biases [49].

6.3 Limitations

The key limitation of this work is the use of a simulated user. Simulations are necessary to systematically explore the space of sampling techniques, search strategies, and tasks we consider in this paper and have been used productively as a complement to human-centered methods [9, 10]. However, simulations will inevitably differ from actual humans and interfaces. We made assumptions about the consistency of users (e.g., that they will choose to or be required to label every document in a batch) and the available features of the simulated interface, and our simulations can neither inform us about users' trust in the interface or resulting classifier nor about users' experience using the system [63]. As a next step, conducting a user study with a prototype IML interface is necessary to verify the patterns we observe via simulation. In particular, we expect iterative search refinement behavior—i.e., a user iteratively refines a query until they achieve the precision they expect from the filtered pool—to have a large impact on estimation bias [19, 39]. Search behavior is hugely affected by the user interface, and early-stage exploration of a dataset is a context where iterative search refinement is particularly useful. Future work for early-stage IML should put humans in the loop to understand when searching is difficult and when search queries help or hurt classifier performance.

We conducted simulations on only a single dataset and constructed tasks based on existing data features. The task of human labeling involves a subjectivity of the concept definition that is not present for the tasks we evaluated. To adopt the language of Chen et al. [16]: while the *data ambiguity* may be high for our tasks—it is hard to determine a product category from a short review without additional context—the *human subjectivity* is low. Additional experiments with labels provided by human annotators and for tasks on which subjectivity is high are necessary. Further, applying our approach to non-text data may better reflect the reality of real-world annotation which often includes multi-modal context for individual data points.

7 CONCLUSION

In this paper, we used simulations to demonstrate trade-offs that affect the design of IML labeling interfaces for text classification. IML labeling interfaces want to minimize human time and effort by collecting labels as efficiently as possible, using methods like active learning and full-text search to enable the user to provide rapid feedback to the classifier. By achieving higher generalization performance with fewer total labels, these methods promise to reduce human cost. However, these methods exist in tension with producing consistent, low-bias estimates of the classifier's performance on the problem of interest. Such estimates enable users to track their progress and estimate how many additional annotations are required to reach a target performance. We showed that "seeding" a classifier using full-text search can increase learning rate without introducing substantial estimation bias. We hope that our findings prompt an interest in a user's initial experience with an IML system during the early-stages of label acquisition, a problem for which sampling and search methods that consider both the needs of the classifier and the user's experience are critical.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful critique and members of the Alexa Shopping research team for useful discussion.

REFERENCES

[1] Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney Tan. 2009. Overview Based Example Selection in End User Interactive Concept Learning. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 247–256. https://doi.org/10.1145/1622176.1622222 event-place: Victoria, BC, Canada.

[2] Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney Tan. 2011. Effective End-User Interaction with Machine Learning. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3736

[3] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09)*. Association for Computing Machinery, New York, NY, USA, 601–610. https://doi.org/10.1145/1645953.1646031

[4] Josh Attenberg and Foster Provost. 2010. Why label when you can search? Alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*. Association for Computing Machinery, New York, NY, USA, 423–432. https://doi.org/10.1145/1835804.1835859

[5] Josh Attenberg and Foster Provost. 2011. Inactive learning? difficulties employing active learning in practice. *SIGKDD Explor. Newsl.* 12, 2 (March 2011), 36–41. https://doi.org/10.1145/1964897.1964906

[6] Katherine Bailey and Sunny Chopra. 2018. Few-Shot Text Classification with Pre-Trained Word Embeddings and a Human in the Loop. *arXiv:1804.02063 [cs]* (April 2018). http://arxiv.org/abs/1804.02063 arXiv: 1804.02063.

[7] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. Association for Computing Machinery, New York, NY, USA, 49–56. https://doi.org/10.1145/1553374.1553381

[8] Alina Beygelzimer, Daniel Hsu, John Langford, and Chicheng Zhang. 2016. Search improves label for active learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 3350–3358.

[9] Yali Bian and Chris North. 2021. DeepSI: Interactive Deep Learning for Semantic Interaction. In *26th International Conference on Intelligent User Interfaces*. ACM, College Station TX USA, 197–207. https://doi.org/10.1145/3397481.3450670

[10] Nadia Boukhelifa, Anastasia Bezerianos, and Evelyne Lutton. 2018. Evaluation of Interactive Machine Learning Systems. *arXiv:1801.07964 [cs]* (Jan. 2018). http://arxiv.org/abs/1801.07964 arXiv: 1801.07964.

[11] Ulisses M. Braga-Neto and Edward R. Dougherty. 2004. Is cross-validation valid for small-sample microarray classification? *Bioinformatics* 20, 3 (Feb. 2004), 374–380. https://doi.org/10.1093/bioinformatics/btg419 Publisher: Oxford Academic.

[12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]* (July 2020). http://arxiv.org/abs/2005.14165 arXiv: 2005.14165.

[13] Carrie J. Cai, Martin C. Stumpe, Michael Terry, Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, and Greg S. Corrado. 2019. Human-Centered Tools for Coping with Imperfect Algorithms During Medical Decision-Making. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, Glasgow, Scotland Uk, 1–14. https://doi.org/10.1145/3290605.3300234

[14] Thiago N. C. Cardoso, Rodrigo M. Silva, Sérgio Canuto, Mirella M. Moro, and Marcos A. Gonçalves. 2017. Ranked batch-mode active learning. *Information Sciences* 379 (Feb. 2017), 313–337. https://doi.org/10.1016/j.ins.2016.10.037

[15] Shuo Chang, Peng Dai, Lichan Hong, Cheng Sheng, Tianjiao Zhang, and Ed H. Chi. 2016. AppGrouper: Knowledge-based Interactive Clustering Tool for App Search Results. In *Proceedings of the 21st International Conference on Intelligent User Interfaces - IUI '16*. ACM Press, Sonoma, California, USA, 348–358. https://doi.org/10.1145/2856767.2856783

[16] Nan-Chen Chen, Margaret Drouhard, Rafal Kocielnik, Jina Suh, and Cecilia R. Aragon. 2018. Using Machine Learning to Support Qualitative Coding in Social Science: Shifting the Focus to Ambiguity. *ACM Trans. Interact. Intell. Syst.* 8, 2

(June 2018), 9:1–9:20. https://doi.org/10.1145/3185515

[17] Rob Chew, Michael Wenger, Caroline Kery, Jason Nance, Keith Richards, Emily Hadley, and Peter Baumgartner. 2019. SMART: An Open Source Data Labeling Platform for Supervised Learning. *Journal of Machine Learning Research* 20, 82 (2019), 1–5. http://jmlr.org/papers/v20/18-859.html

[18] Michael Desmond, Michael Muller, Zahra Ashktorab, Casey Dugan, Evelyn Duesterwald, Kristina Brimijoin, Catherine Finegan-Dollak, Michelle Brachman, Aabhas Sharma, Narendra Nath Joshi, and Qian Pan. 2021. Increasing the Speed and Accuracy of Data Labeling Through an AI Assisted Interface. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 392–401. https://doi.org/10.1145/3397481.3450698

[19] Cecilia di Sciascio, Eduardo Veas, Jordan Barria-Pineda, and Colleen Culley. 2020. Understanding the effects of control and transparency in searching as learning. In *Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI '20)*. Association for Computing Machinery, New York, NY, USA, 498–509. https://doi.org/10.1145/3377325.3377524

[20] Jie Ding, Vahid Tarokh, and Yuhong Yang. 2018. Model Selection Techniques: An Overview. *IEEE Signal Process. Mag.* 35, 6 (Nov. 2018), 16–34. https://doi.org/10.1109/MSP.2018.2867638 arXiv: 1810.09583.

[21] John J. Dudley and Per Ola Kristensson. 2018. A Review of User Interface Design for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 8, 2 (June 2018), 8:1–8:37. https://doi.org/10.1145/3185517

[22] Bradley Efron. 1983. Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. *J. Amer. Statist. Assoc.* 78, 382 (1983), 316–331. https://doi.org/10.2307/2288636 Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

[23] Adrian Englhardt, Holger Trittenbach, Dennis Vetter, and Klemens Böhm. 2020. Finding the Sweet Spot: Batch Selection for One-Class Active Learning. In *Proceedings of the 2020 SIAM International Conference on Data Mining (Proceedings)*. Society for Industrial and Applied Mathematics, 118–126. https://doi.org/10.1137/1.9781611976236.14

[24] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces (IUI '03)*. Association for Computing Machinery, New York, NY, USA, 39–45. https://doi.org/10.1145/604045.604056

[25] George Forman and Martin Scholz. 2010. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.* 12, 1 (Nov. 2010), 49–57. https://doi.org/10.1145/1882471.1882479

[26] Ravi Ganti and Alexander Gray. 2012. UPAL: Unbiased Pool Based Active Learning. In *Artificial Intelligence and Statistics*. PMLR, 422–431. http://proceedings.mlr.press/v22/ganti12.html ISSN: 1938-7228.

[27] Ravi Ganti and Alexander G. Gray. 2013. Building bridges: viewing active learning from the multi-armed bandit lens. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI'13)*. AUAI Press, Arlington, Virginia, USA, 232–241. https://arxiv.org/abs/1309.6830

[28] Aaron Halfaker and R. Stuart Geiger. 2020. ORES: Lowering Barriers with Participatory Machine Learning in Wikipedia. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2 (Oct. 2020), 148:1–148:37. https://doi.org/10.1145/3415219

[29] Blaise Hanczar, Jianping Hua, Chao Sima, John Weinstein, Michael Bittner, and Edward R. Dougherty. 2010. Small-sample precision of ROC-related estimates. *Bioinformatics* 26, 6 (March 2010), 822–830. https://doi.org/10.1093/bioinformatics/btq037 Publisher: Oxford Academic.

[30] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2 Publisher: Springer Science and Business Media LLC.

[31] Neal Harvey and Reid Porter. 2014. User-driven sampling strategies in image exploitation. *Information Visualization* 15, 1 (Nov. 2014), 64–74. https://doi.org/10.1177/1473871614557659

[32] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep Learning Scaling is Predictable, Empirically. *arXiv:1712.00409 [cs, stat]* (Dec. 2017). http://arxiv.org/abs/1712.00409 arXiv: 1712.00409.

[33] Kyle Hipke, Michael Toomim, Rebecca Fiebrink, and James Fogarty. 2014. BeatBox: end-user interactive definition and training of recognizers for percussive vocalizations. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI '14)*. Association for Computing Machinery, New York, NY, USA, 121–124. https://doi.org/10.1145/2598153.2598189

[34] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. (2020). https://doi.org/10.5281/zenodo.1212303

[35] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55 Publisher: IEEE COMPUTER SOC.

[36] Shali Jiang, Roman Garnett, and Benjamin Moseley. 2019. Cost Effective Active Search. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., 4880–4889. https://proceedings.neurips.cc/paper/2019/file/df0e09d6f25a15a815563df9827f48fa-Paper.pdf

[37] Daniel Kottke, Jim Schellinger, Denis Huseljic, and Bernhard Sick. 2019. Limitations of Assessing Active Learning Performance at Runtime. *arXiv:1901.10338 [cs, stat]* (Jan. 2019). http://arxiv.org/abs/1901.10338 arXiv: 1901.10338.

[38] Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 3075–3084. https://doi.org/10.1145/2556288.2557238

[39] Cheng Li, Yue Wang, Paul Resnick, and Qiaozhu Mei. 2014. ReQ-ReC: high recall retrieval with query pooling and interactive classification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (SIGIR '14)*. Association for Computing Machinery, New York, NY, USA, 163–172. https://doi.org/10.1145/2600428.2609618

[40] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent That Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, New York, NY, USA, 577–589. https://doi.org/10.1145/3332165.3347899 event-place: New Orleans, LA, USA.

[41] Christopher H. Lin, Mausam, and Daniel S. Weld. 2018. Active Learning with Unbalanced Classes and Example-Generation Queries. In *Sixth AAAI Conference on Human Computation and Crowdsourcing*. https://www.aaai.org/ocs/index.php/HCOMP/HCOMP18/paper/view/17927

[42] Martin Lindvall and Jesper Molin. 2019. Designing for the Long Tail of Machine Learning. In *arXiv:2001.07455 [cs]*. http://arxiv.org/abs/2001.07455 arXiv: 2001.07455.

[43] Zachary C. Lipton, Yu-Xiang Wang, and Alex Smola. 2018. Detecting and Correcting for Label Shift with Black Box Predictors. *arXiv:1802.03916 [cs, stat]* (July 2018). http://arxiv.org/abs/1802.03916 arXiv: 1802.03916.

[44] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]* (July 2019). http://arxiv.org/abs/1907.11692 arXiv: 1907.11692.

[45] David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2018. Practical Obstacles to Deploying Active Learning. *arXiv:1807.04801 [cs, stat]* (July 2018). http://arxiv.org/abs/1807.04801 arXiv: 1807.04801.

[46] Hao Lü, James A. Fogarty, and Yang Li. 2014. Gesture script: recognizing gestures and their structure using rendering scripts and interactively trained parts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1685–1694. https://doi.org/10.1145/2556288.2557263

[47] Miguel Angel Meza Martínez, Mario Nadj, and Alexander Maedche. 2019. Towards an Integrative Theoretical Framework of Interactive Machine Learning Systems. In *ECIS 2019 proceedings . 27th European Conference on Information Systems (ECIS), Stockholm & Uppsala, Sweden, June 8-14, 2019. Research Papers*. AISeL, Stockholm, Sweden., Paper: 172. https://aisel.aisnet.org/ecis2019_rp/172/

[48] Robert Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI.* Manning Publications.

[49] Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 5356–5371. https://doi.org/10.18653/v1/2021.acl-long.416

[50] Felicia Ng, Jina Suh, and Gonzalo Ramos. 2020. Understanding and Supporting Knowledge Decomposition for Machine Teaching. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference (DIS '20)*. Association for Computing Machinery, New York, NY, USA, 1183–1194. https://doi.org/10.1145/3357236.3395454

[51] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 188–197. https://doi.org/10.18653/v1/D19-1018

[52] Soya Park, April Yi Wang, Ban Kawas, Q. Vera Liao, David Piorkowski, and Marina Danilevsky. 2021. Facilitating Knowledge Sharing from Domain Experts to Data Scientists for Building NLP Models. In *26th International Conference on Intelligent User Interfaces*. ACM, College Station TX USA, 585–596. https://doi.org/10.1145/3397481.3450637

[53] 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. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[54] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389. https://doi.org/10.1561/1500000019

[55] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, and others. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109. Publisher: National Institute of Standards & Technology.

[56] Adam Roegiest and Gordon V. Cormack. 2016. Impact of Review-Set Selection on Human Assessment for Text Classification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 861–864. https://doi.org/10.1145/2911451.2914709

[57] Alvaro Sarasua, Baptiste Caramiaux, and Atau Tanaka. 2016. Machine Learning of Personal Gesture Variation in Music Conducting. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3428–3432. https://doi.org/10.1145/2858036.2858328

[58] Burr Settles. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (June 2012), 1–114. https://doi.org/10.2200/S00429ED1V01Y201207AIM018

[59] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. *arXiv:1805.09843 [cs]* (May 2018). http://arxiv.org/abs/1805.09843 arXiv: 1805.09843.

[60] Patrice Y. Simard, Saleema Amershi, David M. Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and John Wernsing. 2017. Machine Teaching: A New Paradigm for Building Machine Learning Systems. *arXiv:1707.06742 [cs, stat]* (Aug. 2017). http://arxiv.org/abs/1707.06742 arXiv: 1707.06742.

[61] Yunjia Sun, Edward Lank, and Michael Terry. 2017. Label-and-Learn: Visualizing the Likelihood of Machine Learning Classifier's Success During Data Labeling. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces (IUI '17)*. Association for Computing Machinery, New York, NY, USA, 523–534. https://doi.org/10.1145/3025171.3025208

[62] Katrin Tomanek and Udo Hahn. 2008. Approximating Learning Curves for Active-Learning-Driven Annotation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco. http://www.lrec-conf.org/proceedings/lrec2008/pdf/335_paper.pdf

[63] Almar van der Stappen and Mathias Funk. 2021. Towards Guidelines for Designing Human-in-the-Loop Machine Training Interfaces. In *26th International Conference on Intelligent User Interfaces (IUI '21)*. Association for Computing Machinery, New York, NY, USA, 514–519. https://doi.org/10.1145/3397481.3450668

[64] Emily Wall, Soroush Ghorashi, and Gonzalo Ramos. 2019. Using Expert Patterns in Assisted Interactive Machine Learning: A Study in Machine Teaching. In *Human-Computer Interaction - INTERACT 2019 (Lecture Notes in Computer Science)*, David Lamas, Fernando Loizides, Lennart Nacke, Helen Petrie, Marco Winckler, and Panayiotis Zaphiris (Eds.). Springer International Publishing, Cham, 578–599. https://doi.org/10.1007/978-3-030-29387-1_34

[65] Byron C. Wallace, Kevin Small, Carla E. Brodley, Joseph Lau, and Thomas A. Trikalinos. 2012. Deploying an interactive machine learning system in an evidence-based practice center: abstrackr. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium (IHI '12)*. Association for Computing Machinery, New York, NY, USA, 819–824. https://doi.org/10.1145/2110363.2110464

[66] Sida Wang and Christopher Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Jeju Island, Korea, 90–94. https://www.aclweb.org/anthology/P12-2018

[67] Yue Wang, Kai Zheng, Hua Xu, and Qiaozhu Mei. 2017. Clinical Word Sense Disambiguation with Interactive Search and Classification. *AMIA Annu Symp Proc* 2016 (Feb. 2017), 2062–2071. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5333264/

[68] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]* (July 2020). http://arxiv.org/abs/1910.03771 arXiv: 1910.03771.

[69] Steven A. Wolfman, Tessa Lau, Pedro Domingos, and Daniel S. Weld. 2001. Mixed initiative interfaces for learning tasks: SMARTedit talks back. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI '01)*. Association for Computing Machinery, New York, NY, USA, 167–174. https://doi.org/10.1145/359784.360332

[70] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1253–1256. https://doi.org/10.1145/3077136.3080721

[71] Ming Yin, Jennifer Wortman Vaughan, and Hanna Wallach. 2019. Understanding the Effect of Accuracy on Trust in Machine Learning Models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300509

[72] Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start Active Learning through Self-supervised Language Modeling. In *arXiv:2010.09535 [cs]*. http://arxiv.org/abs/2010.09535 arXiv: 2010.09535.

Table 7: References and sections for key concepts

Term	Reference	Description
Interactive machine learning	[24]	Section 2
Active learning	[58]	Section 3.4
Uncertainty sampling	[58]	Section 3.4
Diversity sampling	[72]	Section 3.4
Guided learning	[4]	Section 3.5
Full-text search	[54]	Section 3.5

A KEY CONCEPTS

Table 7 provides references to the section of the paper that describes or operationalizes key concepts in our simulations. In addition, we provide a reference that defines each concept as we use it in this paper.

B FEATURE & LEARNER COMPARISON

We compared true performance at $n = 100$ and $n = 200$ labeled data for various feature representations. In addition to the features used in the main body of the paper (mean-pooled RoBERTa contextualized word embeddings), we compared to bag-of-words (BoW) and TF-IDF feature representations using either unigrams alone or unigrams and bigrams, a common strong baseline in text classification [66]. We tokenized all texts with spaCy's en_core_web_sm [34] and vectorized texts using scikit-learn [53].

Table 8 shows the number of features ($|\beta|$) and the true (test) performance for the various feature representations at $n = 100$ and $n = 200$ randomly-sampled training data. We capped the number of bigram features at 50,000 due to memory restrictions. F1 scores indicate superior performance for RoBERTa embeddings, motivating our default selection of feature representation. Earlier experiments on an alternate sample of Amazon reviews [51] were used to select logistic regression as the default statistical learner—results are shown in Table 9. While the sample is different and we evaluated only on two tasks, sampling regimen and preprocessing were identical to results reported on our primary dataset. Logistic regression was superior or competitive to alternatives at all sample sizes, in addition to being faster to train—a significant benefit in IML contexts. We omit results with multi-layer perceptron learners due to the large search space of hyperparameters and model configurations; results were qualitatively similar to the logistic regression learner but occasionally much worse due to convergence issues

Table 8: True model performance with various feature representations at $n = 100$ and $n = 200$ randomly-sampled training data.

		Features	$\lvert\beta\rvert$	F1	R	P	AP
Is 5 Star	n=100	Unigram BoW	32251	0.7618	0.8245	0.7138	0.7628
		Bigram BoW	50000	0.7803	0.8489	0.7243	0.7899
		Unigram TF-IDF	32251	0.7683	0.9861	0.6300	0.8362
		Bigram TF-IDF	50000	0.7661	1.0000	0.6208	0.8482
		RoBERTa	768	**0.7908**	0.8431	0.7490	0.8409
	n=200	Unigram BoW	32251	0.7909	0.8322	0.7543	0.7945
		Bigram BoW	50000	0.7966	0.8657	0.7379	0.8123
		Unigram TF-IDF	32251	0.7746	0.9658	0.6514	0.8626
		Bigram TF-IDF	50000	0.7732	0.9868	0.6369	0.8726
		RoBERTa	768	**0.8143**	0.8524	0.7811	0.8685
Is Verified	n=100	Unigram BoW	32251	0.8738	0.9645	0.7989	0.7674
		Bigram BoW	50000	0.8726	0.9542	0.8050	0.7840
		Unigram TF-IDF	32251	0.8782	1.0000	0.7828	0.8849
		Bigram TF-IDF	50000	0.8782	1.0000	0.7828	0.8856
		RoBERTa	768	**0.8789**	0.9343	0.8303	0.8857
	n=200	Unigram BoW	32251	0.8744	0.9577	0.8045	0.7953
		Bigram BoW	50000	0.8736	0.9450	0.8128	0.8100
		Unigram TF-IDF	32251	0.8782	1.0000	0.7828	0.8870
		Bigram TF-IDF	50000	0.8781	0.9999	0.7828	0.8890
		RoBERTa	768	**0.8809**	0.9356	0.8326	0.8908
Cat = Books	n=100	Unigram BoW	32251	0.6140	0.4769	0.8901	0.7762
		Bigram BoW	50000	0.5758	0.4197	0.9341	0.7757
		Unigram TF-IDF	32251	0.0013	0.0006	0.6000	0.8163
		Bigram TF-IDF	50000	0.0000	0.0000	0.0000	0.8402
		RoBERTa	768	**0.7458**	0.6521	0.8860	0.8732
	n=200	Unigram BoW	32251	0.7329	0.6009	0.9437	0.8636
		Bigram BoW	50000	0.6944	0.5497	0.9509	0.8517
		Unigram TF-IDF	32251	0.0893	0.0478	0.9988	0.8609
		Bigram TF-IDF	50000	0.0021	0.0010	0.4000	0.8669
		RoBERTa	768	**0.8287**	0.7650	0.9065	0.9188
Cat = Movie/TV	n=100	Unigram BoW	32251	0.1001	0.0766	0.1767	0.0994
		Bigram BoW	50000	0.0960	0.0648	0.2242	0.1229
		Unigram TF-IDF	32251	0.0000	0.0000	0.0000	0.2096
		Bigram TF-IDF	50000	0.0000	0.0000	0.0000	0.2820
		RoBERTa	768	**0.1201**	0.0975	0.2955	0.1960
	n=200	Unigram BoW	32251	0.1626	0.1436	0.3045	0.1572
		Bigram BoW	50000	0.1673	0.1104	0.3765	0.1857
		Unigram TF-IDF	32251	0.0000	0.0000	0.0000	0.2973
		Bigram TF-IDF	50000	0.0000	0.0000	0.0000	0.3384
		RoBERTa	768	**0.2578**	0.2043	0.4203	0.3074

within the training budget at some hyperparameter settings. Table 9 also shows a comparison against alternative RoBERTa embedding strategies: RoBERTa Max uses max pooling rather than mean pooling, while RoBERTa CLS uses the embedding for the CLS token.

C QUERY SURVEY

The query survey was administered via a web form. Five respondents provided the queries shown in Table 10. The following is the survey's text:

To support a broader study, we are collecting data about how humans create text queries to find documents in a category. Imagine that you are searching a database of Amazon reviews using a Google Search-like interface. For each of the six categories below, please

Table 9: True model performance (F1 score) on an alternative data sample for various feature representations and learners. Learners are logistic regression (LogReg), support-vector machines (SVM), and gradient boosting decision tree (GBDT).

		Features	LogReg	SVM	GBDT
Is 5 Star	n=100	BoW (Hashed)	0.788	0.796	0.777
		RoBERTa CLS	0.775	0.758	0.798
		RoBERTa Max	0.778	0.793	0.758
		RoBERTa Mean	**0.815**	0.786	0.809
	n=200	BoW (Hashed)	0.813	0.823	0.775
		RoBERTa CLS	0.786	0.782	0.804
		RoBERTa Max	0.801	0.791	0.773
		RoBERTa Mean	**0.835**	0.799	0.828
Is Verified	n=100	BoW (Hashed)	0.910	0.910	0.911
		RoBERTa CLS	0.910	0.829	0.912
		RoBERTa Max	0.912	0.881	**0.914**
		RoBERTa Mean	0.913	0.828	0.910
	n=200	BoW (Hashed)	0.911	0.909	0.911
		RoBERTa CLS	0.911	0.911	0.909
		RoBERTa Max	0.913	0.892	0.914
		RoBERTa Mean	**0.916**	0.851	0.910

provide 3-10 text queries that you would use to identify reviews in that category. Enter queries one per line or separated with a comma or semicolon. Thank you!

(1) Looking for 5-star reviews. Enter 3-10 text queries in the box below.
(2) Looking for reviews written by Verified Buyers of the product. Enter 3-10 text queries in the box below.
(3) Looking for reviews of books. (Reviews written about products in the Books category.) Enter 3-10 text queries in the box below.
(4) Looking for reviews of movies and TV shows. (Reviews written about products in the Movies & TV category.) Enter 3-10 text queries in the box below.

D PRODUCT CATEGORIZATION

The Amazon Customer Reviews Dataset[11] is segmented into 43 product categories. To match the language of modern categories used on Amazon and improve the validity of the query survey, we combined several of the dataset's product categories to create the category tasks we used in our experiments. The Books category was mapped from categories 'Books' and 'Digital_Ebook_Purchase', while the Movie/TV category was mapped from categories 'Video', 'Video DVD', and 'Digital_Video_Download'. We selected the two categories to reflect a diversity of topic and overall prevalence in the dataset from among categories in common use by Amazon in 2021.

[11]https://s3.amazonaws.com/amazon-reviews-pds/readme.html

E GENERATED QUERIES

In addition to the human-provided queries and the oracle batches, we conducted experiments with two additional sets of generated queries—Broad and Narrow—based on the degree to which they filter the pool and the resulting positive class proportion of the filtered pool. Compared to human-solicited queries, broad and narrow queries both retrieve successively fewer documents in exchange for a greater proportion of positive documents in the filtered pool (see Figure 9). Examples are shown in Table 11. We observed all results involving Broad and Narrow query sets to be qualitatively similar to the Human query set. For example, complete seed search results are shown in Table 12.

Broad query set. From a sampling perspective, the key feature of a query is the positive class proportion in the filtered result pool. For each task, we generated queries by selecting the top ten unigrams that appear in at least 50 training documents and result in the highest positive class proportion. The queries generated for broad search are restricted to return at least 50 documents so that the sampling strategy remains important to the content of the batch when $|B| < 50$.

Narrow query set. In contrast to the broad queries that retrieve many more documents than the batch size, narrow search emulates cases where the searcher is able to generate a query that accurately filters to a high-precision, low-recall subset of documents in the positive class.[12] For each task, we selected the top ten conjunctions (i.e., unigram1 ∧ unigram2) that retrieve *only* positive-class documents and 5-20 documents total.

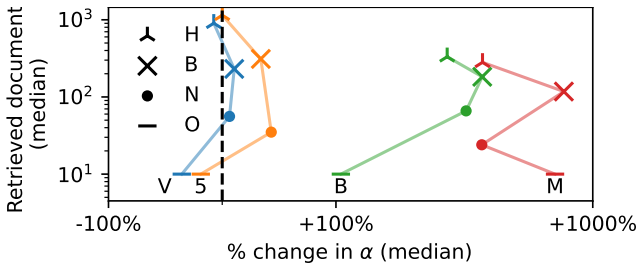

Figure 9: Characteristics of the human (H), broad (B), narrow (N), and oracle (O) query sets. The X axis (log scale) shows the percentage change in the proportion of positive class documents in the unlabeled training pool (α) from the full pool to the query-filtered pool. e.g., Oracle queries always return a filtered pool with α=0.5, which is a decrease in α for Is Verified (V) and Is 5 Star (5) but an increase for Cat = Books (B) and Cat = Movie/TV (M). The Y axis shows the median number of documents retrieved by Anserini for each query.

F UNBIASED POOL-BASED ACTIVE LEARNING WITH LCB-AL

Some active learning approaches use importance sampling to explicitly correct for the sampling bias introduced by using non-random

[12]Generating such queries is implausible in generic search contexts, but an IML interface provides a context where additional conditions can be added to successive queries until the user has identified a query that produces a high positive-class precision.

Table 10: Human-generated queries for each task.

Is Verified	Is 5 Star	Cat = Books	Cat = Movie/TV
Amazon	5-star	Author	Actor
I bought	5-star reviews	Good read	IMDB
I purchased	5-star reviews	an engaging story	Movie
Received it damaged	Buy again	available on kindle	Oscar
Returning the product	Excellent	book	Rotten tomatoes
actual purchase	Great	book	TV
confirmed purchase	High Quality	book recommendation	TV shows
i purchased this several times	Really good	contains practical guidance	a common cliche
i've used it for a long time	Very Happy	detective stories	actress
much better compared to other products	amazing product	fiction	best action movie
my own experience	best	great book	cinema
photo picture	best reviews	negative rating	enjoy watching
verified buyer reviews	buy again	non-fiction	films
verified purchase	exceeds my expectation	novel	genre
verified reviews	excellent quality	overall rating	hilarious episodes
verified reviews	fantastic product	page turner	how xxx

Table 11: Top generated queries for each task, sorted by positive class proportion in the filtered pool.

Task	Broad Queries	Narrow Queries
Is Verified	shoe, iphone, wear, advertised	five∧fit, five∧price, five∧easy, fit∧right
Is 5 Star	five, outstanding, thank, wow	five∧price, five∧fit, five∧easy, five∧best
Cat = Books	readers, author, chapters, book	cant∧reading, story∧got, stars∧reading, reading∧excellent
Cat = Movie/TV	movie, film, watched, dvd	five∧see, didnt∧excellent, product∧story, again∧excellent

Table 12: Seed search with Narrow and Broad query sets. This is an expansion of Table 4. $F1_{n=200}$ for each task and sampling strategy when search seeding is not used, and the change in performance (Δ_{Search}) when search is used in the first batch to seed the classifier.

Task	Query Set	Random		Uncertainty		Diversity	
		$F1_n$	Δ_{Search}	$F1_n$	Δ_{Search}	$F1_n$	Δ_{Search}
Is Verified	Oracle	0.881	-0.001	0.889	-0.001	0.874	-0.004
Is Verified	Human	0.881	+0.001	0.889	-0.000	0.874	-0.000
Is Verified	Broad	0.881	+0.000	0.889	-0.000	0.874	+0.001
Is Verified	Narrow	0.881	+0.001	0.889	-0.001	0.874	+0.001
Is 5 Star	Oracle	0.814	-0.001	0.826	-0.001	0.815	+0.001
Is 5 Star	Human	0.814	+0.000	0.826	-0.000	0.815	-0.000
Is 5 Star	Broad	0.814	+0.001	0.826	-0.003	0.815	+0.002
Is 5 Star	Narrow	0.814	-0.000	0.826	+0.000	0.815	+0.000
Cat = Books	Oracle	0.829	+0.006	0.912	-0.000	0.826	+0.004
Cat = Books	Human	0.829	+0.011	0.912	-0.001	0.826	+0.006
Cat = Books	Broad	0.829	+0.006	0.912	+0.000	0.826	+0.006
Cat = Books	Narrow	0.829	+0.006	0.912	-0.001	0.826	+0.004
Cat = Movie/TV	Oracle	0.258	+0.067	0.667	+0.051	0.166	+0.078
Cat = Movie/TV	Human	0.258	+0.087	0.667	+0.051	0.166	+0.116
Cat = Movie/TV	Broad	0.258	+0.071	0.667	+0.054	0.166	+0.098
Cat = Movie/TV	Narrow	0.258	+0.023	0.667	+0.048	0.166	+0.015

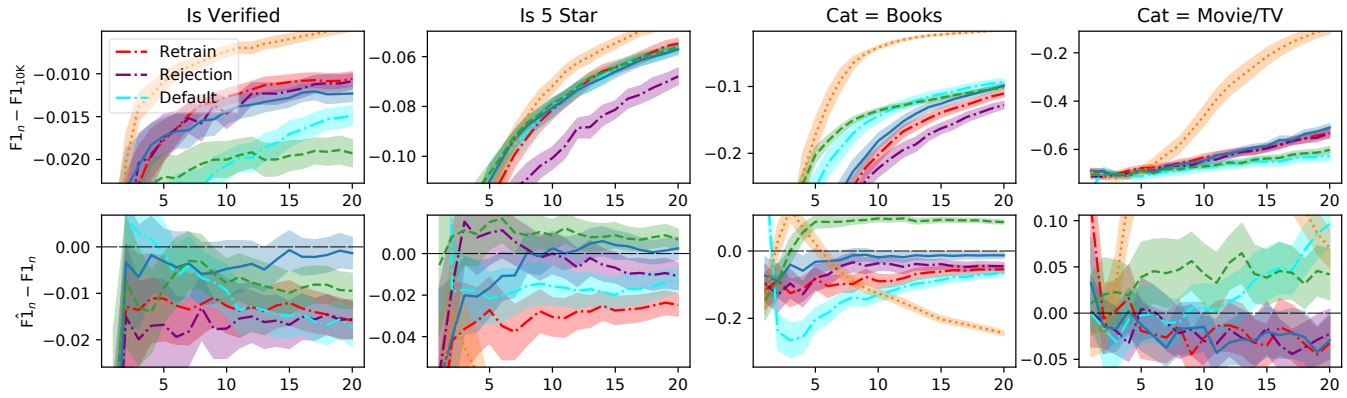

Figure 10: LCB-AL true and estimated performance, added to the existing lines (random, uncertainty, and diversity) from Figure 4. The three estimation strategies are described in the text. This figure is best viewed in color.

sampling [7]. As these approaches provide theoretically-unbiased estimates of the risk [26], we wanted to see if this theory would manifest in practice during early-stage learning and for thresholded metrics such as F1. LCB-AL is the current state-of-the-art among pool-based active learning algorithms that provide these unbiased estimates [27].

We implemented the LCB-AL algorithm, adapted for the batch-mode setting and to disallow re-querying of labeled documents, using PyTorch's L-BFGS implementation to solve the associated optimization problem. As LCB-AL combines sampling and model training into one algorithm, multiple options exist for estimating true performance. We compare three reasonable approaches: (1) **Default**, using the LCB-AL classifier produced after each batch to directly predict on the test set; (2) **Retrain**, throwing out the probability information and LCB-AL-optimized classifier to re-train a new classifier based just on the labeled data, which is the approach used for the other sampling strategies; (3) **Rejection**, using rejection sampling to create a training set from the importance-weighted labeled documents and training a new classifier from the resulting sample, as described by Beygelzeimer et al. [7]. LCB-AL has several hyperparameters, which is another aspect of the approach that makes it ill-fitted for the one-shot IML context. In the results we present here, we set $p_{\min} = \frac{1}{4n}$ and $C_t = \frac{\sqrt{\log(t)}}{100000}$. We found our LCB-AL results to be relatively insensitive to these parameters, although the high values for C_t reported in the original paper caused convergence issues [27]. Results are shown in Figure 10. While LCB-AL's true performance varies by task and which of the three true performance estimation approaches is used, none outperforms random sampling. Generally, the LCB-AL results resemble random sampling in terms of true performance, with additional estimation bias. LCB-AL may perform more effectively outside of the early-stage context, where the high variance involved in the sampling approach provides little benefit.