Toward Interactive Training and Evaluation

Gregory Druck
University of Massachusetts Amherst
gdruck@cs.umass.edu

Andrew McCallum
University of Massachusetts Amherst
mccallum@cs.umass.edu

ABSTRACT

Machine learning often relies on costly labeled data, and this impedes its application to new classification and information extraction problems. This has motivated the development of methods for leveraging abundant prior knowledge about these problems, including methods for lightly supervised learning using model expectation constraints. Building on this work, we envision an interactive training paradigm in which practitioners perform evaluation, analyze errors, and provide and refine expectation constraints in a closed loop. In this paper, we focus on several key subproblems in this paradigm that can be cast as selecting a representative sample of the unlabeled data for the practitioner to inspect. To address these problems, we propose stratified sampling methods that use model expectations as a proxy for latent output variables. In classification and sequence labeling experiments, these sampling strategies reduce accuracy evaluation effort by as much as 53%, provide more reliable estimates of F_1 for rare labels, and aid in the specification and refinement of constraints.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithm, Experimentation, Measurement, Performance

Keywords

interactive training, stratified sampling, evaluation, lightly supervised learning

1. INTRODUCTION

Machine learning often relies on costly labeled data, and this impedes its application to new classification and information extraction problems. However, even in the absence of labeled data we typically have a wealth of prior knowledge about these problems. For example, when extracting information from research papers, we know that the word *ACM* should usually be part of a *journal* or a *conference*. Though this knowledge does not yield labeled data, it does provide a desirable aggregate property of predictions for unlabeled data. Several methods incorporate such properties into learning by encoding them as constraints on the expectations of a probabilistic model [6, 9, 15, 16].

Building on this work, we envision an interactive training paradigm in which users perform evaluation, analyze errors, and specify and refine supervision in a closed loop. In contrast to active learning [21], in this paradigm the system aids the user in understanding what the model is predicting, and the user leverages this insight to direct the learning process. There are at least two benefits to building this paradigm around methods for learning with expectation constraints. First, expectation constraints provide a flexible and powerful language for users to give feedback to the system. Second, interactive analysis may enable users to specify more accurate and useful constraints than would be possible otherwise.

In this paper, we focus on several key subproblems in our interactive training paradigm that can be cast as selecting a representative sample of the unlabeled data for the user to inspect. Specifically, we develop methods to assist the user in performing evaluation at multiple levels of granularity, and in specifying and refining constraints. Random sampling can be applied to these problems, but the resulting sample may not be representative. Instead, we develop sampling strategies based on stratified sampling [24], a method that has a long history in statistics [17]. In stratified sampling, points are grouped into strata, and random sampling is performed within each stratum. If the statistic of interest varies across the strata, then stratified sampling yields a lower variance estimator than random sampling. To perform stratification, we use model expectations as a proxy for latent output variables. In classification and sequence labeling experiments, these sampling strategies reduce accuracy evaluation effort by as much as 53%, provide more reliable estimates of F_1 for rare labels, and aid in the specification and refinement of constraints.

2. RELATED WORK

In contrast to active learning [21], in which the system queries the user, our interactive training paradigm allows the user to provide supervision based on error analysis, which may enable more efficient training. Our interactive paradigm and active learning with expectation constraints [7, 15] could also be combined, allowing a mixture of user-initiated and system-initiated interactions.

There is growing interest in interaction in machine learning, including some work in interactive training and model selection. Huang and Mitchell [10] develop methods for interactive clustering, including allowing the user to provide features that indicate cluster membership, assign instances to clusters, or delete clusters. Roth and Small [18] develop *interactive feature space construction*, in which an annotator uses domain knowledge to iteratively modify model features. Culotta et al. [3] develop a method for interactively correcting errors of Conditional Random Fields (CRFs) [13] using constrained inference. Settles [22] develops an interface for interactive training in which users can label both instances and features. Kumar et al. [12] argue for an interactive classification framework in which annotation cost and both current and future utility are jointly optimized. In contrast, the proposed paradigm is based on learning with expectation constraints[1], and assists the user in evaluation, analysis, and the specification and refinement of constraints.

Note that interactive machine learning often refers to learning directly from interactions with the world, for example based on rewards received for taking context-dependent actions [14]. Though we are interested in exploring this direction in future work, in this paper learning is user-directed.

This paper focuses on developing sampling methods for various subproblems in interactive training. These methods are based on stratified sampling, discussed in Section 4.1, and use model expectations as a proxy for latent output variables. Stratified sampling using model predictions has also been applied to address specific evaluation problems in information retrieval [25]. More closely related, Bennett and Carvalho [1] develop a confidence-based stratified sampling method for estimating classifier accuracy. In contrast, we apply stratified sampling to rapid evaluation of classifiers and structured models trained using expectation constraints. We propose an extension to the method of Bennett and Carvalho [1] that reduces error in this setting in Section 5.1. Additionally, we propose a general approach that is also applicable to other problems in interactive training. Alternatives to stratified sampling include methods based on importance sampling [19, 20]. We plan to consider these approaches in future work.

3. BACKGROUND

In this paper we train discriminative probabilistic models of output variables $\mathbf{y}$ conditioned on observed input variables $\mathbf{x}$. For classification tasks we use logistic regression models, also known as maximum entropy classifiers

$$p(y|\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z(\mathbf{x};\boldsymbol{\theta})}\exp\big(\boldsymbol{\theta}\cdot\mathbf{f}(\mathbf{x},y)\big),$$

where $\boldsymbol{\theta}$ are parameters, $\mathbf{f}$ are model feature functions, and $Z(\mathbf{x};\boldsymbol{\theta}) = \sum_y \exp\big(\boldsymbol{\theta}\cdot\mathbf{f}(\mathbf{x},y)\big)$ is the partition function, which ensures that $\sum_y p(y|\mathbf{x};\boldsymbol{\theta}) = 1$.

For sequence labeling tasks, we use first-order linear chain Conditional Random Fields (CRFs) [13]

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z(\mathbf{x};\boldsymbol{\theta})}\exp\Big(\sum_{t=1}^{T}\boldsymbol{\theta}\cdot\mathbf{f}(\mathbf{x},y_{t-1},y_t)\Big),$$

where $Z(\mathbf{x};\boldsymbol{\theta}) = \sum_{\mathbf{y}}\exp\big(\sum_{t=1}^{T}\boldsymbol{\theta}\cdot\mathbf{f}(\mathbf{x},y_{t-1},y_t)\big)$ is the partition function and T is the sequence length. Inference is performed using the Viterbi and Forward-Backward algorithms. For more detail the reader is referred to [23].

Parameters $\boldsymbol{\theta}$ can be estimated to maximize likelihood if labeled data is available. However, we typically have an abundance of knowledge about the tasks we would like to solve that does not come in the form of labeled instances. For example, when extracting information from research papers, we know that the word *ACM* should usually be part of a *journal* or a *conference*. Recently methods have been developed for estimating parameters with such knowledge and unlabeled data [6, 9, 15, 16]. Formally, these methods encode knowledge as preferences about the values of model expectations of constraint features $\boldsymbol{\phi}(\mathbf{x},\mathbf{y})$. Given unlabeled data $\mathcal{D} = \{\mathbf{x}^1,\ldots,\mathbf{x}^N\}$, the model expectation of $\boldsymbol{\phi}$ is

$$\sum_{j=1}^{N}\mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\theta)}[\boldsymbol{\phi}(\mathbf{x}^j,\mathbf{y})] = \sum_{j=1}^{N}\sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^j;\theta)\boldsymbol{\phi}(\mathbf{x}^j,\mathbf{y}).$$

In this paper we use the Generalized Expectation (GE) framework, though the methods we develop are also applicable to other frameworks such as Posterior Regularization [9]. GE expresses preferences with a score function S that evaluates model expectations of constraint features. A higher score signifies that the model complies with our preferences more closely. To estimate parameters, we solve

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathrm{S}\Big(\sum_{j=1}^{N}\mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\theta)}[\boldsymbol{\phi}(\mathbf{x}^j,\mathbf{y})]\Big) + \log p(\boldsymbol{\theta}).$$

GE has been applied to both logistic regression models [6] and linear chain CRFs [7, 16][2]. GE provides a flexible and powerful framework for users to give feedback to the system during interactive training. Understanding the details of GE is not critical to understanding this paper, so for additional detail the reader is referred to the aforementioned papers.

As in previous work [6, 7, 16], in this paper we predominantly use *input feature label distribution* constraints. For text classification tasks, these constraints specify a target distribution over labels for documents that contain a particular word. For sequence labeling tasks, these constraints specify a target distribution over labels for tokens for which a particular input feature fires. The constraint features $\boldsymbol{\phi}$ are designed so that their expectations are probability distributions, and we use the negative KL divergence from some target distribution $\tilde{\boldsymbol{\phi}}$ as the score function S. These constraints can be obtained by having the user "label" input features, and converting these labels to target distributions using simple heuristics [6, 7, 16]. For example, a user could label *puck* as *hockey*, or *ACM* as *journal* and *conference*.

3.1 Tasks and Data Sets

We conduct text classification and sequence labeling experiments. For classification, the data sets are binary subsets of the *20 Newsgroups* data set used in a previous application of GE to logistic regression [6]. The model features $\mathbf{f}(\mathbf{x},y)$ are word counts, after removing stopwords. To make the experiments in this paper more realistic, we use the input feature label distribution constraints that users provided in the experiments in [6]. We use models trained with these

[1]Settles [22] uses a naive Bayes based method rather than GE (Section 3), but we hypothesize that GE will provide higher accuracy for complex, structured problems, and in settings where constraints have precise target expectations.

[2]Note that an $O(|\mathcal{Y}|^2)$ (rather than $O(|\mathcal{Y}|^3)$ [16]) algorithm has been developed for GE training of linear chain CRFs [5].

constraints as starting points for performing interactive evaluation, error analysis, and refinement. This simulates a scenario in which a user specifies constraints to train an initial model, and then improves the model interactively.

We also conduct experiments with the *Cora* citation extraction data set. The task is to extract 13 BibTex-like fields such as *title* and *journal* from research paper citations. We use a standard set of word, lexicon, and regular expression features, as well as context features in a window of ± 3 tokens [7]. We again use input feature label distribution constraints that were provided by users in previous work [7].

4. SELECTING REPRESENTATIVE SAMPLES

In this paper we aim to develop methods to assist the user in evaluation, error analysis, and the specification and refinement of constraints. We cast these problems as instances of the following more general problem: selecting a sample of the data for the user to inspect. In this section we summarize our *stratified sampling* approach to this problem.

4.1 Stratified Sampling

We first review *stratified sampling* [24], a sampling method we use pervasively in this paper. We assume an iid unlabeled dataset $\mathcal{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$. We aim to compute the mean of a per-instance function $r(\mathbf{x}^j, \mathbf{y}^j)$ that considers both the input variables and the true values of the latent output variables. To simplify notation in this section we define $r^j = r(\mathbf{x}^j, \mathbf{y}^j)$. The true mean is $\bar{r}^* = \frac{1}{N}\sum_{j=1}^{N} r^j$. Because the output variables $\mathbf{y}^j$ are latent, evaluating r is costly. Rather than evaluating r for each instance, we choose a sample S of size n and use it to estimate the mean of r.

The most basic strategy is *random sampling*, in which n instances are selected uniformly at random from $\mathcal{D}$. The estimate of the population mean is the sample mean $\hat{\bar{r}}_{rs} = \frac{1}{n}\sum_{j=1}^{n} r^j$. This estimator is unbiased, meaning that the expected value of $\hat{\bar{r}}_{rs}$ over all possible samples of size n is $\bar{r}^*$. However, when the sample size n is small, $\hat{\bar{r}}_{rs}$ has high variance. With replacement[3], the estimated variance of $\hat{\bar{r}}_{rs}$ is $\hat{V}ar(\hat{\bar{r}}_{rs}) = \frac{S^2}{n}$, where S^2 is the sample variance.

Stratified sampling has a long history in statistics [17]. In stratified sampling, instances are partitioned into m strata $\{\mathcal{D}_{s1}, \ldots, \mathcal{D}_{sm}\}$, where each $x^j \in \mathcal{D}$ appears in exactly one stratum. To obtain a complete sample of size n, for each stratum i, n_i instances are randomly sampled ($n = \sum_{i=1}^{m} n_i$). An estimate of the mean of r can be obtained using

$$\hat{\bar{r}}_{ss} = \sum_{i=1}^{m} \frac{N_i}{N} \frac{1}{n_i} \sum_{j=1}^{n_i} r_i^j = \sum_{i=1}^{m} W_i \hat{\bar{r}}_i,$$

where r_i^j is r for the jth instance in the ith stratum, $W_i = N_i/N$ is the *weight* of the ith stratum, and $\hat{\bar{r}}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} r_i^j$ is the mean estimate for the ith stratum. This estimator is also unbiased. The estimated variance of $\hat{\bar{r}}_{ss}$ is

$$\hat{V}ar(\hat{\bar{r}}_{ss}) = \sum_{i=1}^{m} W_i^2 \hat{V}ar(\hat{\bar{r}}_i),$$

where $\hat{V}ar(\hat{\bar{r}}_i) = S_i^2/n_i$ is the estimated variance of $\hat{\bar{r}}_i$. A $1-\alpha$ confidence interval for $\hat{\bar{r}}_{ss}$ is $\hat{\bar{r}}_{ss} \pm z_{\alpha/2}\sqrt{\hat{V}ar(\hat{\bar{r}}_{ss})}$.

[3]Without replacement $\hat{V}ar(\hat{\bar{r}}_{rs}) = (1 - \frac{n}{N})\frac{S^2}{n}$.

For a given sample size n, we consider two methods for choosing the number of samples from each stratum. With *proportional allocation*, the sampling proportions are equal to the weights, $n_i/n = W_i$. It can be shown that the true variance of the proportional allocation estimator is lower than the true variance of the random sampling estimator: $Var(\hat{\bar{r}}_{ss}) \leq Var(\hat{\bar{r}}_{rs})$. The difference of the variances is

$$Var(\hat{\bar{r}}_{rs}) - Var(\hat{\bar{r}}_{ss}) = \frac{1}{n}\sum_{i=1}^{m} W_i(\bar{r}_i^* - \bar{r}^*)^2. \qquad (1)$$

Equation 1 has implications in stratification, as it shows that the variance reduction is larger when strata means $\bar{r}_i^*$ have high variance, or are very different from each other.

The second sample allocation method is *optimal allocation.* In optimal allocation, per-stratum sample sizes n_i are not proportional to the size of the stratum. Instead, the idea is to use more samples in strata where r has higher variance. Formally, suppose that the true standard deviations σ_i for each stratum are known. Optimal allocation assigns samples to each stratum using the following equation

$$n_i = n \times \frac{N_i \sigma_i}{\sum_{i'=1}^{m} N_{i'}\sigma_{i'}}.$$

Optimal allocation can outperform proportional allocation when the variance of r varies across different strata.

Because the r^j are latent, we must select proxy variables to stand in for them to perform stratified sampling. These proxy variables can be used for both stratification and estimating strata variances for optimal allocation.

4.2 Estimating Vector-Valued and Non-Linear Functions

In many applications of interest we need to estimate the means of several functions simultaneously, which we view as estimating the mean of a vector-valued function $\mathbf{r}$ instead of a scalar function r. Often a vector-valued function $\mathbf{r}$ is required because we are interested in estimating a *non-linear* function with the sample. A non-linear function is one that cannot be computed as the mean of a function on individual instances in the sample. For example, accuracy is the mean of a function that evaluates correctness on each instance, but F_1 is non-linear, as computing it requires the number of correct, predicted, and true instances in the entire sample. Vector-valued and non-linear functions have implications in stratification, sample allocation, and estimation.

Stratification & Sample Allocation: As described in Section 4.1, for scalar r strata should be selected so that stratum means $\bar{r}_i^*$ are much different than the population mean $\bar{r}^*$. In the vector-valued $\mathbf{r}$ case, stratification is less straightforward. Intuitively, the stratification should be helpful for estimating all elements of $\mathbf{r}$, but note that for some non-linear functions certain estimates r_i may be more important than others. We simply stratify so that a composite variable, correlated with $\mathbf{r}$, varies across strata. Alternative methods for future study include multi-way [2] and clustering-based stratification. We also perform optimal allocation using the estimated standard deviation of a composite variable.

Estimation: A natural estimator of a non-linear function ω is $\hat{\omega} = \omega(\hat{\bar{\mathbf{r}}})$ [11]. Note that although $\hat{\bar{\mathbf{r}}}$ is an unbiased estimate of $\bar{\mathbf{r}}^*$, $\hat{\omega}$ is not necessarily an unbiased estimate of $\omega(\bar{\mathbf{r}}^*)$. Computing the variance of $\hat{\omega}$ is not straightforward, as we only obtain one value of ω from the sample. A general approach is to use re-sampling methods such as jackknifing, in which each instance is held out of the sample in turn, providing n different function values that can be used to estimate variance [8]. Jackknife estimates for stratified sampling have also been developed [11].

4.3 General Stratified Sampling Approach

For each stratified sampling method we propose throughout this paper, we describe four components: the *target function*, the *stratification function*, the *stratification scheme*, and the *sample allocation scheme*. We next describe these components and our general approach to designing them.

As described in the previous section, in some cases we are not interested in the mean estimates $\hat{\bar{\mathbf{r}}}$, but rather some non-linear function of the estimates $\omega(\hat{\bar{\mathbf{r}}})$. We refer to the non-linear function ω as the *target function*. The target function varies in different applications. In the linear case the target function is the identity function.

The *stratification function* s computes a proxy value for each instance that can be used for stratification and sample allocation. Though the output variables $\mathbf{y}$ are latent, if we have a trained model we do have some information about the values of these variables. Our general approach is to use current model predictions as a proxy for latent output variables $\mathbf{y}$. Specifically, a method we use pervasively is to stratify according to the *expectation* of some function of interest r' (often $r' = r$)

$$s(\mathbf{x}^j) = \mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta})}[r'(\mathbf{x}^j, \mathbf{y})] = \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta}) r'(\mathbf{x}^j, \mathbf{y}).$$

This approach makes the reasonable assumption that model predictions are correlated with the true output variables. This implies that the improvement provided by stratified sampling is bounded by accuracy of the current model.

When estimating a vector of means $\hat{\bar{\mathbf{r}}}$, we take a simple approach and continue to use a scalar stratification function $s(\mathbf{x})$, essentially defining a composite variable. One simple strategy is to use the expectation of a single element of the vector: $r' = r_i$, $s(\mathbf{x}) = \mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta})}[r_i(\mathbf{x}^j, \mathbf{y})]$.

The *stratification scheme* takes as input the values computed by the stratification function and defines the strata. We use two methods for defining strata, which we discuss in detail in the following sections.

Finally, the *sample allocation scheme* defines how samples are allocated to strata. If optimal allocation is used, rather than proportional allocation, then a *stratum variance estimator* that defines a procedure for estimating the variance of strata must also be defined. We estimate within-stratum variances by using the sample estimates (for instances that have already been inspected), using model probabilities as a proxy for the latent output variables, or a combination. To use model probabilities, we advocate a simple strategy in which values for the latent output variables are sampled according to the model, r is computed, and the resulting sample variance estimates of r stand-in for true variances.

Note that, in general, inspecting individual instances to estimate a statistic could also yield labeled instances as a by-product. In future work we could perform training using both constraints and labeled instances.

5. OVERALL EVALUATION

Suppose we have an initial model for the task we would like to solve. Our goal is to interactively improve this model. A natural first step is to come to an understanding of what the model is predicting and how accurate it is, in order to decide where to focus analysis and provide supervision. Importantly, we want to do this with minimal effort. In this section we apply stratified sampling to choose sets of instances (with model predictions) for manual inspection. For concreteness and to allow thorough validation, we focus on the task of using the sample to estimate a performance metric such as accuracy. However, we also suggest that with appropriate adjustments to the sampling scheme, viewing a sample of instances selected by this method would provide accurate representations of other properties of interest.

Note that the methods described here, though motivated by interactive training, could also be used in a non-interactive setting to evaluate lightly supervised learning. This is an important problem on its own, as in a lightly supervised setting there is typically no data available for evaluation.

5.1 Evaluating Classification Accuracy

We first estimate classification accuracy. Formally, we aim to estimate the mean of the correctness indicator function

$$r_c(\mathbf{x}, y) = 1_{\{\hat{y}=y\}},$$

where $\hat{y}$ is the predicted label, $\hat{y} = \mathrm{argmax}_y\, p(y|\mathbf{x};\boldsymbol{\theta})$, and $1_{\{p\}}$ returns 1 if the predicate p is true, and 0 otherwise. When computed with the true label, r_c returns 1 if the model is correct, and 0 otherwise. We next devise several stratified sampling schemes for estimating $\hat{\bar{r}}_c$.

Target Function: Classification accuracy is a linear function, as $\hat{\bar{r}}_c$ is an estimate of accuracy. Consequently the target function ω is simply the identity function $\omega(\hat{\bar{r}}_c) = \hat{\bar{r}}_c$.

Stratification Function: Equation 1 suggests that strata should be defined to maximize the variance in individual stratum accuracies. Because the stratum accuracies are unavailable, we instead stratify using the expectation of r_c.

$$\begin{aligned} s_c(\mathbf{x}^j) &= \mathrm{E}_{p(y|\mathbf{x}^j;\boldsymbol{\theta})}[r_c(\mathbf{x}^j, y)] \\ &= \sum_y p(y|\mathbf{x}^j;\boldsymbol{\theta}) 1_{\{\hat{y}=y\}} = p(\hat{y}|\mathbf{x}^j;\boldsymbol{\theta}) \end{aligned} \tag{2}$$

Equation 2 shows that the model expectation of r_c is the probability of the best label, or the model's *confidence* in its prediction. Note that, using our general approach, we recover the stratification function of Bennett and Carvalho [1]. When applied to other tasks in interactive training our approach yields different stratification functions.

Stratification Scheme: We use a *uniform size stratification scheme*. First, we sort unlabeled instances according to $s_c(\mathbf{x}^j)$. Then, we define strata by splitting the sorted list into m pieces, each containing the same number of instances.

Sample Allocation: Finally, we must allocate samples to the strata. With a uniform size stratification scheme, *proportional allocation* allocates n/m samples to each stratum. We additionally use *optimal allocation*, where the challenge is estimating the standard deviations in each stratum $\hat{\sigma}_i$.

Bennett and Carvalho [1] propose *online stratified sampling*, in which the $\hat{\sigma}_i$ are re-estimated using the observed values r_i^j after each sample. As the number of samples increases, the estimates become more accurate, and savings increase. However, a potential disadvantage of this approach is that many samples may be required to obtain estimates that are accurate enough to be beneficial. Because we are especially interested in evaluation with minimal effort, we propose two additional methods for estimating $\hat{\sigma}_i$ that leverage model predictions.

We can model each unobserved r_i^j in stratum i as a Bernoulli random variable c_i^j with $p_i^j = p(\hat{y}|\mathbf{x}^j;\boldsymbol{\theta})$. Summing all c_i^j for stratum i yields an expected accuracy random variable with a *Poisson Binomial* distribution[4]. We can use the variance of this distribution as an estimate of the stratum variance $\hat{\sigma}_i^2 = \sum_j p_i^j(1-p_i^j)$. This method prioritizes strata where there are expected to be a mix of correct and incorrect predictions over strata with expected accuracy near 0 or 1.

As n increases, we expect *online stratified sampling* to eventually provide better estimates than the above method. Consequently, we propose a novel compromise. In each iteration, we sample a correctness value for each unlabeled instance $c_i^j \sim p(\mathbf{y}^j|\mathbf{x}^j;\boldsymbol{\theta})$, and treat these values as pseudo-observations of r_i^j that are down-weighted by parameter α. We then estimate $\hat{\sigma}_i$ as in *online stratified sampling*, combining the true and pseudo-observations.

5.2 Classification Experiments

We compare approaches for evaluating the accuracy of document classifiers for binary subsets of *20 Newsgroups* (see Section 3.1). The classifiers are trained with GE using constraints provided by users in previous work [6].

We compare random sampling (*random*) and stratified sampling approaches that use confidence stratification with $m = 5$ strata and different sample allocation methods: proportional allocation (*pro conf*), optimal allocation using online variance estimation (*opt online*) [1], and optimal allocation using the combined confidence and online variance estimation method (*opt conf online*). We also conduct but do not display experiments with confidence-based optimal allocation. In general, *opt conf online* outperforms this method more as n increases. We begin stratified sampling by allocating two samples to each stratum. For the optimal allocation methods, we reestimate $\hat{\sigma}_i$ after each sample, and smooth the estimates with 10 pseudo-observations[5]; for *opt online* these pseudo-observations are uniform, while for *opt conf online* the pseudo-observations are sampled correctness values (i.e. $\alpha = 10/N_i$). To compute estimates of $\hat{\bar{r}}$, we reveal the true labels for instances in the sample. We run 1000 trials, and report the mean absolute accuracy estimation error.

Figure 1 displays error vs. n. First, note that the stratified sampling approaches provide lower mean absolute error than random sampling. We assess significance with a Mann-Whitney U test, the non-parametric counterpart of an unpaired t-test. Of the 216 possible comparisons between random sampling and a stratified sampling method (9 tasks $\times$ 8 different sample sizes $\times$ 3 stratified sampling methods), stratified sampling provides significantly lower error (significance level $\alpha = 0.05$) in 209 cases. Random sampling never significantly outperforms stratified sampling. Attaining the same mean absolute error with random sampling would often require significantly more effort. For example, in the *med-space* task with *User 1*'s constraints, a sample of size $n = 20$ using *opt conf online* gives error of 0.0406. *Random* attains comparable performance with 30 samples, giving an error of 0.0418. This is a 33% reduction in evaluation effort.

[4]The sum is a Binomial random variable if p is the same for each instance.

[5]We initially smoothed with $1/\sqrt{n_i}$ pseudo-observations as in [1], but this significantly increased error for *opt online.*

We conclude that the accuracy of classifiers trained with GE can be estimated more efficiently using stratified sampling.

Opt conf online provides lower mean absolute error than any other method in 54 of the 72 cases (9 tasks $\times$ 8 sample sizes). Of the 162 reductions in these 54 cases, 133 are significant. *Opt conf online* significantly outperforms *opt online* 43 times, and is significantly outperformed by *opt online* only once. Error analysis reveals that *opt online* tends to overestimate the differences in variance between strata. Additional smoothing reduces error with large n, but increases error with small n, as *opt online* approaches *pro conf* as the amount of smoothing increases. We conclude that *opt conf online* is preferable for minimal effort evaluation.

5.3 Estimating Token Accuracy

We next propose stratified sampling methods for evaluating token accuracy for sequence labeling models. We estimate the mean of the vector-valued function $\mathbf{r}_{sc}$ defined

$$r_{sc0}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} 1_{\{\hat{y}_t = y_t\}}$$
$$r_{sc1}(\mathbf{x},\mathbf{y}) = T,$$

where T is the length of the sequence and t indexes positions in the sequence. The function $\mathbf{r}_{sc}$ returns the number of correctly predicted labels in the first position, and the length in the second. We describe the components of several stratified sampling schemes.

Target Function: While instance accuracy and average token accuracy are linear, token accuracy is non-linear. Token accuracy is defined $\omega_{ta}(\hat{\bar{\mathbf{r}}}_{sc}) = \hat{\bar{r}}_{sc0}/\hat{\bar{r}}_{sc1}$[6].

Stratification Functions: We propose two stratification functions. The first is the expectation of r_{sc0}.

$$s_{exc}(\mathbf{x}^j) = \mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta})}[r_{sc0}(\mathbf{x}^j,\mathbf{y})] = \sum_{t=1}^{T} p(\hat{y}_t|\mathbf{x}^j;\boldsymbol{\theta})$$

This can be interpreted as the expected number of correct tokens, where $p(\hat{y}_t|\mathbf{x}^j;\boldsymbol{\theta})$ is the marginal probability of the predicted label at position t. The stratification function s_{conf} is the expectation of a function that returns 1 only if the labeling of the entire sequence is correct.

$$s_{conf}(\mathbf{x}^j) = \mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta})}[1_{\{\hat{\mathbf{y}}=\mathbf{y}\}}] = p(\hat{\mathbf{y}}|\mathbf{x}^j;\boldsymbol{\theta}).$$

The expectation is the probability of the predicted label sequence. This can be interpreted as model confidence.

Stratification Scheme: We use the *uniform size stratification scheme*, described in Section 5.1, for this task.

Sample Allocation: We use both proportional and optimal allocation. In optimal allocation, we allocate samples using estimates of the standard deviation of $r_{sc0}(\mathbf{x}^j,\mathbf{y})$. This is preferable to using the standard deviation of $1_{\{\hat{\mathbf{y}}=\mathbf{y}\}}$ in the applications we explore here, as complete instance accuracy is low. As in Section 5.1, we use both online estimation of the variance with the samples, and a combined method. In the combined method, pseudo-observations of token correctness are sampled according to the marginal

[6]We could use the population mean length in place of $\hat{\bar{r}}_{sc1}$. Stratified sampling provides even larger improvements with this estimator, but the estimates have higher error due to discrepancies between the population mean length and $\hat{\bar{r}}_{sc1}$. This may result in accuracy >1, for example. Therefore, in this paper we estimate all arguments to ω from S.

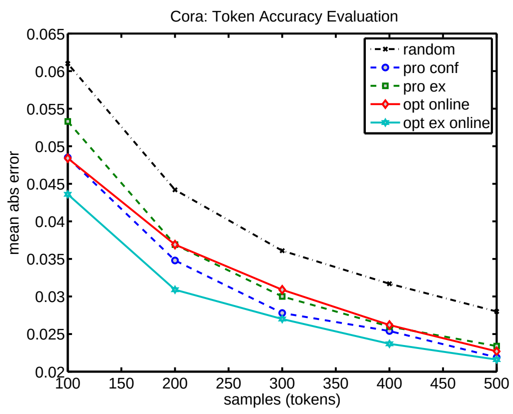

Figure 2: Stratified sampling methods significantly outperform random sampling for evaluating token accuracy on the *Cora* data set.

probabilities of the predicted labels $p(\hat{y}_t|\mathbf{x}^j;\boldsymbol{\theta})$, and these pseudo-observations are combined with the labeled sample estimates, as described in Section 5.1.

5.4 Sequence Labeling Experiments

This experiment uses the *Cora* data set as described in Section 3.1. The model is trained with the constraints provided by User 3 in [7], and an additional constraint with weight 10 that specifies that 80% of transitions between labels should be self-transitions. In this task each sequence is a complete citation. To enable precise sampling, we split citations into smaller subsequences. However, using very short subsequences would obscure helpful context, making the user's task more difficult. Consequently, we split citations into subsequences of maximum length 10.

The sampling strategies for this experiment are *random*, s_{conf} stratification with proportional allocation (*pro conf*), s_{exc} stratification with proportional allocation (*pro ex*), s_{conf} stratification with online optimal allocation (*opt online*), and s_{conf} stratification with combined optimal allocation (*opt ex online*). We do not report results with s_{exc} stratification and optimal allocation. The results are similar to *opt ex online*, but with higher error. For *opt online*, we used the same smoothing strategy as [1], which performed better than a fixed value of 10 for this task (the opposite was true in Section 5.2). We did not tune α for *opt ex online*, keeping $\alpha = 10/N_i$, giving an advantage to *opt online*. We use $m = 5$ strata, and conduct 1000 trials. To simulate rapid evaluation, we use 100-500 tokens ($\sim$10-50 subsequences).

Figure 2 displays the results. The x-axis is the total number of tokens evaluated. All stratified sampling methods significantly outperform random sampling (Mann Whitney U test with significance level $\alpha = 0.05$). *Opt ex online* outperforms all methods significantly at each point on the graph except for *pro conf* at 300, and *pro conf* and *opt online* at 500. At 200 tokens, the mean absolute error with *opt ex online* is 0.0309. *Random* sampling does not attain a comparable error of 0.0311 until $n = 425$. This is a savings of 225 tokens, or 53% of total evaluation effort.

In this experiment *pro conf* outperforms *pro ex*. Note that $s_{exc} \leq T$. Error analysis shows that s_{exc} yields strata that are more correlated with T than s_{conf}.

6. FINE-GRAINED EVALUATION

In Section 5, we found that stratified sampling methods can provide significantly more accurate estimates of overall performance than random sampling. One possible next step in interactive training is to drill down and perform fine-grained evaluation, with the goal of using this information to refine or provide new supervision. In this section, we propose a stratified sampling approach to fine-grained evaluation and error analysis. As an example error analysis task, we consider computing token F_1 for a particular label of interest ℓ for sequence labeling tasks. This is an important problem because in a particular application some labels may be more important than others. Additionally, awareness of low F_1 for ℓ provides a path for improving the model.

The function of interest, $\mathbf{r}_{f\ell}$, is defined as

$$r_{f\ell 0}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} 1_{\{y_t=\ell \,\wedge\, \hat{y}_t=\ell\}}$$

$$r_{f\ell 1}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} 1_{\{\hat{y}_t=\ell\}}$$

$$r_{f\ell 2}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} 1_{\{y_t=\ell\}}.$$

In words, the first element is the number of correctly predicted labels whose value is ℓ, the second element is the number of predictions of ℓ, and the third element is the number of true labels whose value is ℓ.

Target Function: The F_1 target function for label ℓ is

$$\omega_{F_1}(\hat{\bar{\mathbf{r}}}_{f\ell}) = \frac{2 \times \hat{\bar{r}}_{f\ell 0}/\hat{\bar{r}}_{f\ell 1} \times \hat{\bar{r}}_{f\ell 0}/\hat{\bar{r}}_{f\ell 2}}{\hat{\bar{r}}_{f\ell 0}/\hat{\bar{r}}_{f\ell 1} + \hat{\bar{r}}_{f\ell 0}/\hat{\bar{r}}_{f\ell 2}}.$$

Stratification Function: The stratification function is the expectation of $r_{f\ell 2}$

$$s_{f\ell 2}(\mathbf{x}^j) = \mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta})}[r_{f\ell 2}(\mathbf{x}^j,\mathbf{y})] = \sum_{t=1}^{T} p(y_t = \ell|\mathbf{x}^j;\boldsymbol{\theta}).$$

This can be interpreted as the expected number of ℓ tokens.

Stratification Scheme: We again use the *uniform size stratification scheme*. However, we find that the density of $r_{f\ell 2}$ is less uniform than the density of the stratification functions used for overall evaluation. There are often a few $\mathbf{x}^j$ for which $s_{f\ell 2}$ is large, and many $\mathbf{x}^j$ for which $s_{f\ell 2}$ is very small. This occurs when ℓ is infrequent, for example.

Consequently, we also experiment with non-uniform size stratification. A standard method for determining stratum boundaries is the *cum* $\sqrt{F}$ *rule* [4], which aims to minimize within-stratum variance. Performing stratification according to the cum $\sqrt{F}$ rule first involves splitting the instances into k initial sorted classes, where C_i denotes the ith class. Next, the cumulative function c of the square root of the class frequencies is computed.

$$c(j) = \sum_{i=1}^{j} \sqrt{|C_i|}$$

The *stratum width* w is then computed as $w = c(k)/m$. Finally, the initial classes are grouped into strata of equal width w using the cum $\sqrt{F}$ values, $c(j)$.

Sample Allocation: We allocate two initial samples to each stratum (to enable computing $\hat{V}ar(\hat{\bar{r}}_{ss})$ if needed), and allocate the remaining samples proportionally. When $n = 2m$, this can be viewed as optimal allocation with $\hat{\sigma}_i$ that are inversely proportional to the stratum sizes, $\hat{\sigma}_i = 1/N_i$. This scheme is appropriate because in this setting large strata are likely to have small $s_{f\ell 2}$ values, and consequently we expect few occurrences of ℓ in those strata.

6.1 Experiments

We use the same data set and initial model as in Section 5.4. Citations are again split into subsequences of maximum length 10. To simulate obtaining a rapid estimate of token F_1 for ℓ, we evaluate using $n = 10$ subsequences.

We compare *random* sampling, sampling proportionally from equal-sized strata using $s_{f\ell 2}$ (*ex uniform*), and sampling proportionally from strata determined by the cum $\sqrt{F}$ rule using $s_{f\ell 2}$ (*ex cum*$\sqrt{F}$). We use $m = 5$ strata. For the *cum* $\sqrt{F}$ rule, we use $k = 20$ initial classes that each cover a fixed-width segment of the range of $s_{f\ell 2}$ (i.e. one class is all $\mathbf{x}$ with $s_{f\ell 2}(\mathbf{x}) \in [0, 0.5]$). In some cases with $k = 20$ initial classes less than m of them contain instances — empirical evidence of the statement above that the density of $s_{f\ell 2}$ can be highly non-uniform. In this case we multiply k by 10 iteratively until we have at least $m + 1$ non-empty classes.

Table 1 reports the results. We evaluate both the mean absolute error in the F_1 estimate (F_1 *err*) and the percentage of wasted trials (*waste*). A wasted trial occurs when none of the 10 subsequences contain a true occurrence of the label of interest. In this case it is not possible to obtain a meaningful recall estimate[7]. Bold denotes that a method gives the lowest F_1 *err* or *waste*. A * in the F_1 *err* column denotes statistical significance (Mann Whitney U test with significance level $\alpha = 0.05$).

The *ex cum*$\sqrt{F}$ method performs as well as or better than the other methods. In terms of F_1, it always significantly outperforms *random*, and significantly outperforms *ex uniform* in all cases except *title* and *date*. The *ex cum*$\sqrt{F}$ method also avoids a wasted sample in 1000 trials in all cases except for *note*. Using non-uniform strata with $s_{f\ell 2}$ typically results in a small stratum with instances that are very likely to contain ℓ. This greatly reduces waste. However, stratified sampling also samples other instances, ensuring that we obtain an unbiased estimate of $\bar{\mathbf{r}}^*$ even if model predictions are poorly correlated with the true labels. This illustrates the utility of stratified sampling for targeted evaluation.

	random		ex uniform		ex cum$\sqrt{F}$	
	F_1 err	waste	F_1 err	waste	F_1 err	waste
author	0.073	0.004	0.051	**0.000**	**0.041***	**0.000**
journal	0.273	0.069	0.221	0.006	**0.143***	**0.000**
note	0.843	0.843	0.822	0.822	**0.274***	**0.274**
booktitle	0.183	0.017	0.139	**0.000**	**0.105***	**0.000**
tech	0.392	0.501	0.363	0.364	**0.129***	**0.000**
volume	0.293	0.084	0.239	0.004	**0.091***	**0.000**
location	0.286	0.286	0.262	0.145	**0.093***	**0.000**
editor	0.510	0.605	0.468	0.488	**0.172***	**0.000**
institut.	0.351	0.427	0.339	0.257	**0.136***	**0.000**
title	0.079	**0.000**	0.056	**0.000**	**0.051**	**0.000**
date	0.123	0.006	0.080	**0.000**	**0.077**	**0.000**
pages	0.183	0.052	0.133	**0.000**	**0.081***	**0.000**
publisher	0.392	0.149	0.356	0.071	**0.157***	**0.000**

Table 1: Using $n = 10$ subsequences to evaluate token label F_1 for each *Cora* label. Stratified sampling provides more accurate F_1 estimates and avoids wasted samples.

7. SPECIFYING NEW CONSTRAINTS

Thus far we have focused on evaluation and analysis. We now shift our focus to improving the model. Users could specify new constraints manually, or select from candidate constraints [6, 7]. In this section we propose a new paradigm in which the user specifies constraints while inspecting data. For example, after inspecting [$\underline{\textit{journal}}$: Transactions] [$\underline{\textit{title}}$: on Pattern Analysis] ..., the user may choose to add new constraints that specify that *Transactions* should almost always be labeled *journal*, and that transitions from *journal* to *title* are extremely unlikely. This paradigm can help the user specify more accurate constraints, find constraints that are particularly useful, and may suggest constraints to the user that they may not have considered otherwise. Note that constraints apply to the entire data set, and hence provide more supervision than labeling data [6, 7]. However, in future work we plan to allow both types of supervision. We focus on *targeted improvement* of the model. In this section we aim to improve token F_1 for a particular label ℓ.

We can view this as an estimation problem as follows. Based on their prior knowledge, the user has some set of candidate constraints that they are capable of specifying. For each instance, the function $\mathbf{r}$ simply returns the number of times each candidate constraint is applicable with respect to the targeted improvement task. Note that computing $\mathbf{r}$ is expensive, since the user must manually inspect instances. The user decides which constraints to add based on the mean candidate constraint estimate $\hat{\bar{\mathbf{r}}}$. When applying stratified sampling to improve label ℓ, we use the same sample allocation and cum $\sqrt{F}$ stratification method as in Section 6.

7.1 Experiments

We use the same data set and constraints as in Section 5.4. Initial models are trained with either the *full* set of 108 constraints or a random subsample of 52 constraints (*small*).

We simulate the specification of new constraints so that we can conduct a large number of trials. To simulate user prior knowledge, we use labeled data to define constraints as in previous work [6, 7, 15]. Candidate constraints include input feature label distribution constraints of the form used in [7] for input features that occur at least 10 times and have label distribution entropy ≤ 0.7. In addition, there are candidate constraints that discourage unlikely label transitions[8]. In this experiment unlikely transitions are those that do not occur in the labeled data. Candidate constraints are applicable if they apply to a token with true or predicted label ℓ. Any constraint i with $\hat{\bar{r}}_i \geq 0$ is then added. For input feature label distributions, the target distribution is assigned using the "labeling" method used in [7].

We evaluate the targeted improvement of three moderately infrequent labels in the *Cora* data set: *institution*, *journal*, and *location*. We use $n = 10$ sub-sequences of length at most 10 to find constraints, and subsequently retrain the model with the augmented set of constraints. Results com-

[7]Following standard conventions, recall is 1 if there are no true occurrences of ℓ in the sample, and precision is 1 if there are no predicted occurrences of ℓ in the sample.

[8]The probability of taking any transition in the unlikely set is encouraged to be close to 0 using KL divergence. To balance this constraint with the others, its weight is set to the number of labeled feature constraints.

constraints	label	initial	random	stratified
small	institution	0.178	0.271	**0.448***
	journal	0.341	0.473	**0.634***
	location	0.466	0.580	**0.684***
full	institution	0.661	0.682	**0.717***
	journal	0.635	0.593	**0.671***
	location	0.655	0.685	**0.727***

Table 2: Using stratified sampling to find new constraints improves token F_1 for a label of interest.

paring *random* and *stratified* sampling with 100 trials are presented in Table 2. In all cases, stratified sampling yields significantly higher token F_1 for ℓ. The improvement is the result of finding additional applicable constraints. While specifying these constraints takes additional time, when ℓ is infrequent, we expect finding appropriate constraints to dominate the time required to specify them. As in Section 6, this method encourages the sample to contain a mix of correct predictions of ℓ, false positives, and false negatives. Other sampling strategies do not provide this coverage. For example, uncertainty sampling may miss true occurrences of ℓ, and certainty sampling method may miss false negatives.

8. ESTIMATING TARGET EXPECTATIONS

Input feature label distribution constraints typically use target distributions over labels that are set with simple heuristics [6, 7, 16]. It is known that as these target distributions become more precise, the resulting model becomes more accurate [16]. Additionally, users occasionally make mistakes when specifying constraints. For example, in the *baseball-hockey* task, User 1 mistakenly provided the constraint that *Devils* $\mapsto$ *baseball*. The user was likely thinking of the Tampa Bay Devil Rays, rather than the New Jersey *Devils* (a hockey team). Such incorrect constraints can be detrimental to GE.

Incorrect constraints can be corrected and imprecise constraints can be refined by having the user view a few occurrences of the constraint feature ϕ in context. Mann and McCallum [16] found that this target estimation method gave higher accuracy than traditional sequence labeling with the same number of labels. In this section we use stratified sampling to ensure that the small number of occurrences considered by the user are representative. Note that these ideas could be applied to other expectation estimation problems.

The specific task we consider is estimating a distribution over labels for a particular input feature q. The function of interest returns a vector with the count of q with each label. For a classification task, the function $\mathbf{r}_q$ is

$$r_{q\ell}(\mathbf{x}, y) = 1_{\{y=\ell\}} q(\mathbf{x})$$

For a sequence labeling task, the function $\mathbf{r}_q$ is

$$r_{q\ell}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} 1_{\{y_t=\ell\}} q(\mathbf{x}, t)$$

Target Function: We aim to estimate the distribution over labels for each input feature. Consequently $\omega_{ex}(\hat{\bar{\mathbf{r}}}_q)$ simply returns $\hat{\bar{\mathbf{r}}}_q$ normalized to sum to 1.

Stratification Functions: We need only consider $\mathbf{x}^j$ where $q(\mathbf{x}^j)=1$, or $q(\mathbf{x}^j, t)=1$ for some t. For binary classification tasks, the stratification function is the expectation of r_{q0}, where 0 refers to one of the labels.

$$s_{q0}(\mathbf{x}^j) = \mathrm{E}_{p(y|\mathbf{x}^j;\boldsymbol{\theta})}[r_{q0}(\mathbf{x}^j, y)] = p(y = 0|\mathbf{x}^j; \boldsymbol{\theta}),$$

constraint and data sets	random		stratified	
	err	rt acc	err	rt acc
(1) baseball-hockey	0.178	0.932	**0.113***	**0.948***
(1) ibm-mac	0.274	0.784	**0.244***	**0.790***
(1) med-space	0.139	0.921	**0.108***	**0.931***
(2) baseball-hockey	0.234	0.912	**0.163***	**0.932***
(2) ibm-mac	0.310	**0.782**	**0.303***	**0.782**
(2) med-space	0.205	0.912	**0.175***	**0.919***
(3) baseball-hockey	0.178	0.923	**0.119***	**0.942***
(3) ibm-mac	0.250	0.809	**0.226***	**0.816***
(3) med-space	0.112	0.922	**0.094***	**0.928***

Table 3: Stratified sampling provides lower error target expectation estimates, and higher accuracy when the classifier is retrained with the refined targets.

where again we only consider $\mathbf{x}^j$ with $q(\mathbf{x}^j)=1$.

For a non-binary task we stratify according to the index of the most frequently predicted label ℓ_{max}.

$$\begin{aligned} s_{\ell_{max}}(\mathbf{x}^j) &= \mathrm{E}_{p(\mathbf{y}|\mathbf{x}^j;\boldsymbol{\theta})}[r_{q\ell_{max}}(\mathbf{x}^j, \mathbf{y})] \\ &= \sum_{t=1}^{T} p(y_t = \ell_{max}|\mathbf{x}^j; \boldsymbol{\theta}) q(\mathbf{x}^j, t). \end{aligned}$$

In ongoing work we are developing improved, clustering-based methods for stratification for expectation estimation.

Stratification Scheme and Sample Allocation: We use the *cum* $\sqrt{F}$ *rule*, described in Section 6, for stratification, and the same sample allocation scheme.

8.1 Classification Experiment

For this experiment we use the same data sets and constraints as the experiments in Section 5.2. We use $m = 2$ strata, 4 samples per constraint, and repeat the experiment 1000 times with different random seeds. We evaluate using the mean absolute expectation estimation error (*err*), and the accuracy of the logistic regression model after re-training with the refined constraints (*rt acc*). Table 3 displays results comparing *random* sampling and stratified sampling with cum $\sqrt{F}$ rule stratification using s_{q0} (*stratified*). Stratified sampling always provides more accurate expectation estimates, and provides higher accuracy when the model is retrained with the refined constraints in all cases except one. Cases in which stratified sampling significantly outperforms random sampling are indicated with a *.

8.2 Sequence Labeling Experiment

Finally, we refine User 3's constraints for *Cora* with $n=4$ and $n = 10$ samples of the constraint occurring in context. We use $m=2$ strata for $n=4$, $m=5$ strata for $n=10$, and conduct 100 trials. We use the same initial model as in Section 5.4, which has accuracy of 82.8%. Note that here strata with low $s_{\ell_{max}}$ values do not necessarily have low variance. Therefore, in this experiment we avoid over-stratifying, allowing $< m$ strata if there are fewer non-empty initial classes.

Using random sampling gives mean absolute expectation estimation error of 0.259 with $n=4$ and error of 0.171 with $n=10$. Using proportional allocation with s_{qmax} and cum $\sqrt{F}$ stratification gives error of 0.215 with $n=4$, a 17% error reduction, and error of 0.136 with $n=10$, a 20% error reduction. Retraining the model with the refined constraints obtained using *random* sampling gives accuracy of 82.5% with $n = 4$ and accuracy of 86.3% with $n = 10$, while retraining the model with refined constraints using stratified sampling gives statistically significantly higher accuracy of 84.0% with $n=4$ and accuracy of 87.2% with $n=10$. Random sampling requires $n=16$ samples per constraint to match the accuracy of stratified sampling with $n=10$, a 37.5% reduction.

The user may instead use the sample to specify their own target expectation. In this case, we want the sample to include as many labels as possible, to remind the user of the input feature's uses. Random sampling with $n=4$ finds 61.1% of the labels input features occur with, whereas stratified sampling finds 70.4%. With $n=10$, random sampling finds 69.5%, whereas stratified sampling finds 82.9%.

We also conjecture that in applications where the target label distributions have higher entropy, reductions in target estimation error will yield larger accuracy improvements.

9. FUTURE WORK

In this paper we addressed subproblems in interactive training that can be cast as selecting a representative sample for the user to review. In this section we discuss future opportunities in interactive training. In addition to the directions below, we are interested in stratification schemes that additionally consider the input variables, and in developing a complete interactive training system.

Model Prediction Rationales: In addition to helping the user understand what the model is predicting, it may be beneficial to help the user understand why the model is making a prediction. For example, knowing why the model is making a mistake could help a user add the necessary constraints to fix it. We conjecture that a user could understand a model prediction by inspecting similar contexts and studying the way the prediction changes with the input.

Summarizing Differences between Models: We are interested in developing sampling methods for summarizing the similarities and differences between two models. This could help the user understand how the model is changing with the addition of new constraints.

Detecting Incorrect Constraints: In some cases we suspect that incorrect constraints could be automatically detected, as we have observed that constraints whose targets are poorly matched during training are often incorrect. This method could assist the user in prioritizing refinement.

10. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, and the University of Massachusetts gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the view of DARPA, AFRL, or the US government.

11. REFERENCES

[1] P. N. Bennett and V. R. Carvalho. Online stratified sampling: evaluating classifiers at web-scale. In *CIKM*, pages 1581–1584, 2010.

[2] D. J. H. C. J. Skinner and D. Holt. Multiple frame sampling for multivariate stratification. *International Statistical Review*, 62(3):333–347, 1994.

[3] A. Culotta, T. Kristjansson, A. McCallum, and P. Viola. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence*, 170(14):1101–1122, 2006.

[4] T. Dalenius and J. Hodges. Minimum variance stratification. *J. Amer. Statist. Ass.*, 30(219-229), 1959.

[5] G. Druck. *Generalized Expectation Criteria for Lightly Supervised Learning.* PhD thesis, University of Massachusetts Amherst, 2011.

[6] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR*, 2008.

[7] G. Druck, B. Settles, and A. McCallum. Active learning by labeling features. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 81—90, 2009.

[8] B. Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans.* CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial Mathematics, 1987.

[9] K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049, July 2010.

[10] Y. Huang and T. M. Mitchell. Text clustering with extended user feedback. In *SIGIR*, pages 413–420, 2006.

[11] D. Krewski and J. N. K. Rao. Inference from stratified samples: Properties of the linearization, jackknife and balanced repeated replication methods. *The Annals of Statistics*, 9(5):1010–1019, 1981.

[12] M. Kumar, R. Ghani, M. Shah, J. G. Carbonell, and A. I. Rudnicky. Framework for interactive classification problems. In *ICML Workshop on Combining Learning Strategies to Reduce Label Cost*, 2011.

[13] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int'l. Conf. on Machine Learning*, pages 282–289, 2001.

[14] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2007.

[15] P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. In *Proc. Int'l. Conf. on Machine Learning*, pages 641–648, 2009.

[16] G. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proc. of Meeting of Assoc. for Computational Linguistics*, pages 870–878, 2008.

[17] J. Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[18] D. Roth and K. Small. Interactive feature space construction using semantic information. In *CoNLL*, 2009.

[19] C. Sawade, N. Landwehr, S. Bickel, and T. Scheffer. Active risk estimation. In *Proceedings of the International Conference on Machine Learning*, 2010.

[20] C. Sawade, N. Landwehr, and T. Scheffer. Active evaluation of f-measures. In *Advances in Neural Information Processing Systems.* 2010.

[21] B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin - Madison, 2009.

[22] B. Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478, 2011.

[23] C. Sutton and A. Mccallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning.* MIT Press, 2006.

[24] S. K. Thompson. *Sampling.* Wiley-Interscience, 2002.

[25] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 603–610, 2008.

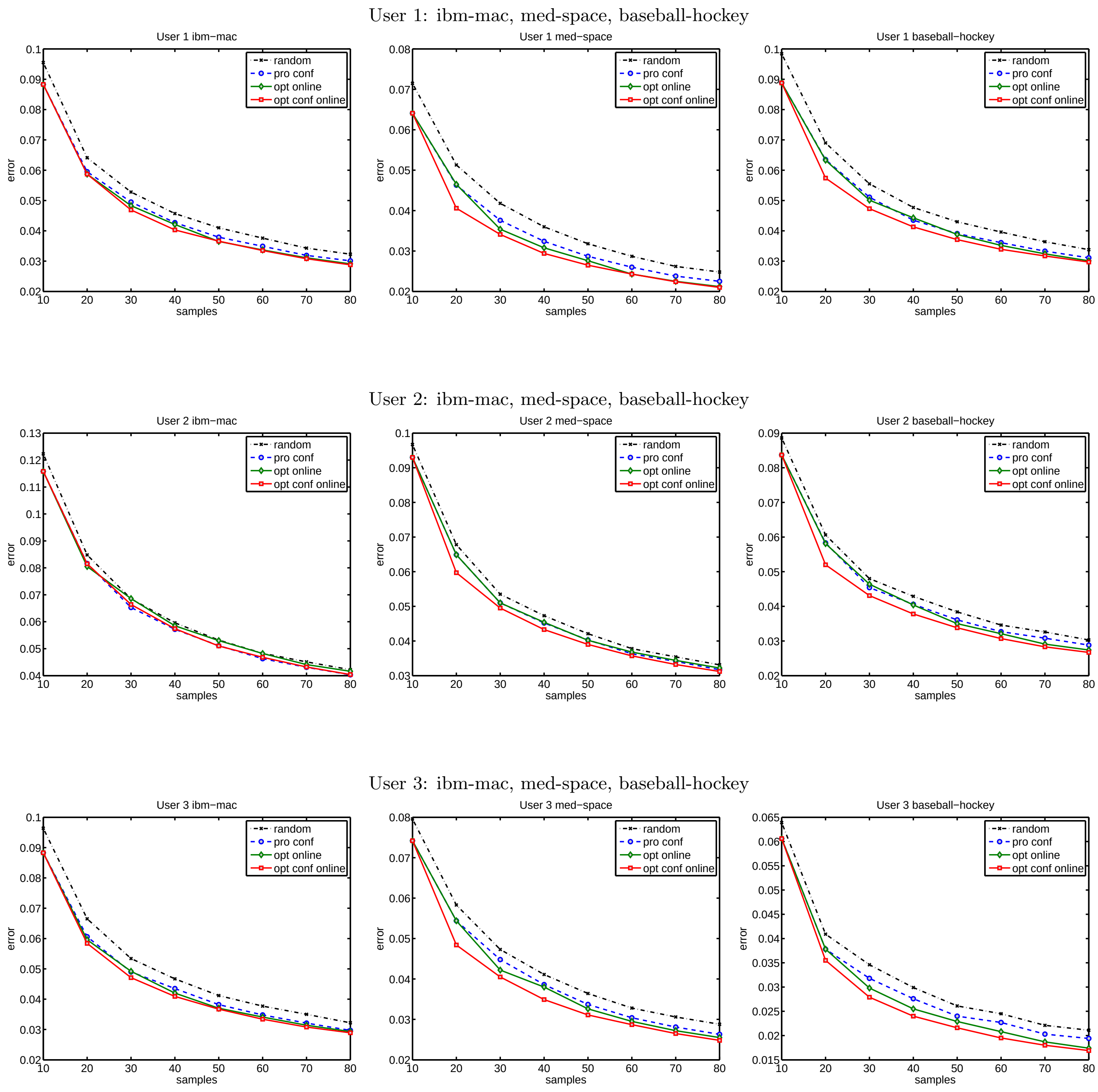

Figure 1: Stratified sampling methods provide classification accuracy estimates with lower error. ***Opt conf online*** **typically outperforms all other methods.**