Towards Rapid Interactive Machine Learning

Evaluating Tradeoffs of Classification without Representation

Dustin Arendt
Pacific Northwest National Laboratory
dustin.arendt@pnnl.gov

Emily Saldanha
Pacific Northwest National Laboratory
emily.saldanha@pnnl.gov

Ryan Wesslen
University of North Carolina at Charlotte
rwesslen@uncc.edu

Svitlana Volkova
Pacific Northwest National Laboratory
svitlana.volkova@pnnl.gov

Wenwen Dou
University of North Carolina at Charlotte
wdou1@uncc.edu

ABSTRACT

Our contribution is the design and evaluation of an interactive machine learning interface that rapidly provides the user with model feedback after every interaction. To address visual scalability, this interface communicates with the user via a "tip of the iceberg" approach, where the user interacts with a small set of recommended instances for each class. To address computational scalability, we developed an $O(n)$ classification algorithm that incorporates user feedback incrementally, and without consulting the data's underlying representation matrix. Our computational evaluation showed that this algorithm has similar accuracy to several off-the-shelf classification algorithms with small amounts of labeled data. Empirical evaluation revealed that users performed better using our design compared to an equivalent active learning setup.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; *User studies*; • **Computing methodologies** → *Machine learning algorithms*;

KEYWORDS

Interactive Machine Learning; Representation-Free Classifier; Visual Interactive Labeling; Active Learning; Transduction Learning; Hierarchical Clustering

ACM Reference format:
Dustin Arendt, Emily Saldanha, Ryan Wesslen, Svitlana Volkova, and Wenwen Dou. 2019. Towards Rapid Interactive Machine Learning. In *Proceedings of 24th International Conference on Intelligent User Interfaces, Marina del Rey, CA, USA, March 17–20, 2019 (IUI '19),* 12 pages.
https://doi.org/10.1145/3301275.3302280

IUI '19, March 17–20, 2019, Marina del Ray, CA, USA

ACM ISBN 978-1-4503-6272-6/19/03…$15.00
https://doi.org/10.1145/3301275.3302280

1 INTRODUCTION

The past decade has seen better models, more powerful hardware, and higher quality training data lead to significant advancements in supervised machine learning, i.e., learning an arbitrary function from examples [44]. These methods are dependent on large amounts of high-quality labeled data for training, which can be very difficult to obtain. Effective training sets are available for some application domains like handwriting [34] and image recognition [18, 29]. Crowdsourcing platforms can presumably be used to curate large training sets like these when expertise is prevalent in the general population, or when expertise is easily transferrable. However, this "army of annotators" approach is not always feasible or cost effective. We believe this to be the rule rather than the exception, because there are many situations where only a very small number of domain experts, e.g., scientists or analysts, have the knowledge to label instances. Thus, the problem we address here is *label elicitation*: building an effective training set for a supervised classifier by acquiring labels from a domain expert.

Active Learning (AL) is a well established label elicitation framework for finding more accurate classifiers with less budget spent, i.e., queries made of the user [48]. A key feature of AL systems is that an algorithm ranks instances by their potential benefit to the classifier if labeled. Generally instances are presented to the user serially, and the algorithm updates its ranking after each user input. In practice, domain experts prefer more autonomy, dislike playing the role of the oracle, and want to receive a benefit from the system that exceeds the cost of the time spent [1].

Visual Interactive Labeling (VIL) also addresses the problem of acquiring labels from a few domain experts [8]. In contrast to AL, VIL systems are typically exploratory data analytics tools that put the user in control of what instances to label. VIL tools sometimes allow the user to understand the distribution of instances in a high dimensional feature space, e.g., through dimension reduction, to help speed up the labeling process. AL and VIL appear to be on opposite ends of a spectrum of user control. With AL the algorithm is in control, the user's task is well-defined, and the interface is fairly simple; with VIL the user is in control, the user's task is open ended, and the interface is fairly complex.

We propose an approach for label elicitation that lies between AL and VIL on this spectrum, borrowing aspects from both techniques. This allows users to explore and label the data simultaneously, which is especially useful when classes are not known *a priori*. Our

approach is implemented in a web-based tool called CHISSL and our contributions in this paper are the following:

- a tabular "tip of the iceberg" design that is visually scalable, preventing users from becoming overwhelmed;
- an $O(n)$ transduction classification algorithm that is computationally scalable, enabling rapid feedback; and
- a computational and empirical evaluation showing that our system is fast, accurate, and helpful.

Our contributions were motivated by the speed and accuracy limitations of off-the-shelf supervised and semi-supervised classification algorithms that we experienced during preliminary research into this problem [3].

CHISSL's "tip of the iceberg" design selects a small subset of the most relevant instances to show the user, and arranges them into rows by predicted class. Instances are are not abstracted and are allocated significant screen space, e.g., images are shown natively and not transformed into dots on a scatter plot. The user directly manipulates instances with drag and drop, to provide labels for the model. The model provides immediate feedback after every interaction by updating the few instances shown for each group. Our proposed $O(n)$ transduction classification algorithm is *representation free*—it incrementally incorporates user feedback without referring to the underlying representation (feature) matrix. This is accomplished with a 1-Nearest Neighbor approach that uses a dendrogram produced by hierarchical clustering to approximate distance between instances.

Despite an expected tradeoff between speed and accuracy, our computational evaluation showed that with few labels, our algorithm can be as accurate, but faster than off-the-shelf semi-supervised algorithms. We also conducted a user study revealing that participants were more accurate using CHISSL compared to AL. *Representation-free* algorithms are well suited for interactive machine learning interfaces, and our work lays the groundwork for future research in this area.

This paper is organized as follows. In the next section, we introduce the terminology surrounding supervised and semi-supervised classification we use throughout the paper. We also discuss existing VIL and AL techniques and their limitations in this context. Following this we present our design constraints, leading directly to our algorithmic and user interface contributions. The remainder of the paper evaluates these designs, first from a computational standpoint, and second from a usability standpoint. In our discussion, we interpret the results of this evaluation, discuss limitations and other considerations of the overall approach, and present lessons learned that lead to immediate design improvements.

2 BACKGROUND & RELATED WORK

Supervised classification is a branch of machine learning where the goal is to learn an arbitrary function that can predict the label $y \in \mathbb{Z}^+$ of an arbitrary instance $\mathbf{x} \in \mathbb{R}^n$ given a finite set of observed $(\mathbf{x}, y)$ pairs [44]. We refer to the observations used to train the model as $\mathbf{X}_{train}$ and $\mathbf{y}_{train}$. A subset of the observations are withheld from the training and used for validation only, and we refer to these as $\mathbf{X}_{test}$ and $\mathbf{y}_{test}$. While *supervised classification* has received much recent attention, the *semi-supervised* variant of this problem is relevant to label elicitation.

In this case, there are missing labels for some, or even most of the observed instances. We designate which of our data corresponds to the labeled L or unlabeled U components, $\mathbf{X}_L$ and $\mathbf{X}_U$, and $\mathbf{y} = \mathbf{y}_L$. The supervised problem is to fit a model given $\mathbf{X}_L$ and $\mathbf{y}_L$, whereas the *semi-supervised* problem is to use $\mathbf{X}_L$, $\mathbf{X}_U$, and y_L. Thus, the intuition behind *semi-supervised learning* is that the unlabeled representation can provide some benefit to the model by providing additional observations that further characterize the feature space [53].

Supervised classification, as we have described it, performs *induction learning* because models can predict the label of arbitrary observations. Alternatively, some models are restricted to *transduction learning*, where they only predict on data provided at the same time as training. Thus, a *semi-supervised* classifier that trains with $\mathbf{X}_L$, $\mathbf{X}_U$, and $\mathbf{y}_L$ and only predicts on $\mathbf{X}_U$ performs transduction learning.

These problem formulations more or less appropriate depending on the context and available data. For example, with Active Learning (AL) we may only have labels available once they are provided directly by the user, and it is unreasonable to assume that we will eventually have labels for all collected instances. So, *semi-supervised learning* or *transduction learning* may be effective depending on whether we want to build a predictive model, or whether we simply wish categorize an existing data set.

We characterize some existing *semi-supervised* classifiers as *representation-free*, which we define as algorithms that transform the representation matrix into a more compact data structure, enabling training and prediction without the original feature matrix. Some graph-based semi-supervised methods [53] perform classification directly on a graph structure generated from the representation matrix, making them *representation-free*. For example, Blum and Chawla proposed a technique where labeled nodes are treated as sources and sinks, and the graph min-cut is used to classify unlabeled instances [11]. The remainder of this section builds on this problem setup and focuses on visual analytics approaches related to the problem of organizing or labeling large datasets.

2.1 Interactive Clustering

Some interactive clustering techniques allow the user to adjust the parameters of the clustering algorithm [47], However, we review approaches that leverage constrained clustering [6, 10] as a means for the user to interact with the clustering algorithm, which helps ensure that the clustering matches the user's domain knowledge and expectation of quality. There are primarily two types of constraints elicited from the user: split/merge and rank/accept/reject. Split/merge constraints [4, 5, 16, 35] allow the user to dictate whether two clusters belong together, or whether one cluster should actually be two. This result can also be achieved using pairwise constraints on data points [19].

Using rank/accept/reject constraints, the user can provide feedback about how well a given cluster matches what she is looking for. For example, Kwon et al. generate many clusters using different algorithms and parameter settings and let the user filter out those that are not acceptable [33]. Alternatively, because it may be too expensive to generate many clusters, Srivasta et al. developed a system where the clustering algorithm takes into account clusters

previously rejected by the user [49]. Spatialization is a related technique where the system learns the user's mental model by observing the user manipulation of a few instances and uses this information to learn a user-tailored projection of all instances [13, 22, 27]. We mention these techniques, because they can implicitly cluster the data by separating semantically related sets of instances into visually distinct clusters.

2.2 Visual Interactive Labeling

A user-driven alternative to AL, Visual Interactive Labeling (VIL) [8, 9] allows the user to select which instances to label and uses a classifier to predict unlabeled instances' classes. VIL has been used to support classification of images [8, 23, 39, 41, 42, 46], video [26], documents [25], web-pages [31], and social media user profiles [2]. Several interactive clustering techniques could reasonably be characterized as VIL approaches, because they allow the user to directly label instances, to assign instances to clusters, or to specify instances as cluster seeds [12, 14–16].

Most of the above techniques visualize instances as points projected into a 2-D space using traditional dimension reduction techniques like principal components analysis [28], multidimensional scaling [30], or neighbor embeddings like t-SNE [38]. These techniques help the user to visually evaluate cluster quality or pick important instances by helping her understand the density distribution of instances and their classes. However, this approach has clear usability and scalability limitations. Projected data can be misleading [17] or difficult to interpret, limiting the application of many of these tools to experts such as data scientists. Additionally, outliers in the projected space that draw the user's attention may not be statistical outliers [52]. Typically, projected instances are abstracted as colored circles because of the sheer quantity of points needed to convey the relevant information. However, this can be counter-productive for VIL where the purpose is to allow the user to quickly identify mis-labeled instances, because the relevant information has been abstracted away.

3 DESIGN

Amershi et al. proposed that effective interactive machine learning systems should be "rapid, focused, and incremental" [1]. To the best of our knowledge, none of the aforementioned techniques simultaneously address all of these guidelines. Thus, our first goal was to develop a rapid, focused, and incremental label elicitation tool that performs on datasets typical for the data scientists and researchers we collaborate with at our institution. These datasets are typically 10-100K instances across diverse data types such as text, images, time series, event sequences, and graphs. Our target was to incorporate user feedback within the 100ms rule of thumb [37], so that updated predictions could be rapidly provided to the user after each interaction.

Our second goal was to design an interface that benefitted from both AL and VIL techniques and provided the user a level of control between these two approaches. With these goals in mind, we designed CHISSL within a set of constraints, identified *a priori*, intended to mitigate threats to scalability and generalizability. The latter part of this section presents our design of the user interface and algorithms and discusses how they address our constraints.

3.1 Constraints for Scalability

Below are the four constraints that guided our design of a rapid interactive machine learning system. We designed with a web-based application in mind, but note that many of the threats we identified are not exclusive to this architecture. Two of the guidelines are computational (**CC1** and **CC2**) and two are visual (**VC1** and **VC2**).

3.1.1 **(CC1)** *Don't fit or predict on the server.*

Threat: Latency. High client-server latency could degrade the responsiveness of the system if the user must wait for a server response after every interaction. This often occurs when the client and server are not within the same network.

Threat: Surging Demand. Multiple concurrent users can cause challenges when the server is expected to provide realtime analytics. If the server does not have the capacity handle demand spikes, then the responsiveness of the system degrades. It is expensive to maintain an architecture capable of this, and it is desirable to perform client-side computation when reasonable.

3.1.2 **(CC2)** *But, don't send the representations to client, either.*

Threat: Load time. Representation matrices can be quite large in practice, e.g., for images there can be thousands of floating point values per instance. Transmitting the entire representation matrix up front is annoying to the user, requiring her to wait for it to download before she can begin.

Threat: Browser limitations. Web browsers have limited resources—running compute intensive methods can cause the browser to be unresponsive. The representation matrix could exceed the memory capacity of the browser.

3.1.3 **(VC1)** *Don't aggregate instances.*

Threat: Over-abstraction. VIL assumes the user can effectively select helpful instances to label for the model, but this is impeded if instances are abstracted, e.g., as points in a dimension reduction plot. If the user cannot see instance data without additional interaction, then the benefits from pre-attentive visual search are lost.

Threat: Over-specialization. Different data should aggregate in different ways, showing individual instances avoids *ad hoc* aggregation and interaction approaches.

3.1.4 **(VC2)** *But, don't render every instance either.*

Threat: Clutter and over-plotting. There is simply not enough screen space to render every instance except when datasets are trivially small. Even scatter plots suffer from over-plotting and clutter with large datasets.

3.2 Algorithm Design

Our analysis pipeline is divided into two phases: 1) a slow batch phase, and 2) a fast interactive phase. Our server-side batch phase is typical of most machine learning applications, where instances are processed into an n instances by m features representation matrix, $\mathbf{X}$. Once a feature matrix is created, our classification algorithm is domain agnostic. The final step of the batch phase is to apply Ward agglomerative clustering [40, 50], which returns a sequence

of pairwise merges. However, any hierarchical clustering algorithm can, in principle, be used to produce a dendrogram.

To address **CC2**, only the hierarchy is passed to the client, which performs classification exclusively using this structure, i.e., without **X**. We add weights to the edges in the hierarchy as free parameters which can further improve the classification accuracy. CHISSL predicts labels for all instances using 1-nearest neighbor (1-NN) classification, where the weighted path distance in this hierarchy approximates the distance between instances. Figure 1 illustrates this concept. The hierarchy requires $O(n)$ bytes of memory, which we note can be smaller than **X**, which requires $O(n \cdot m)$ bytes. An $O(n)$ implementation of our 1-NN classifier, details on selecting borderlines and suggestions, and an $O(n \cdot m)$ heuristic for tuning hierarchy weights are detailed in the Appendix.

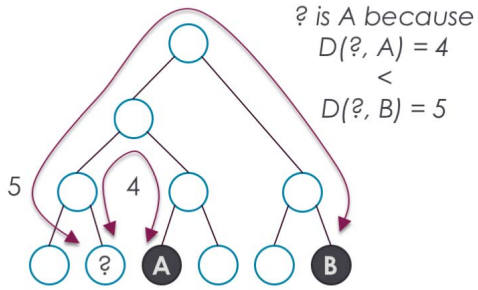


Figure 1: Classification without representation—This diagram shows how the output of hierarchical clustering is used for transduction learning. Instances are leaf nodes in the bottom level of the drawing, and two instances have been labeled: one with class "A", the other with class "B". The unlabeled instance "?" is predicted as class "A" because it is closer than "B".

3.3 Interface Design

We designed our interface to be generalizable, avoiding solutions that reveal details or inner-workings of the clustering or classification algorithm. For example, many techniques exist for visualizing hierarchical data [21], but we consider such approaches off limits given our design assumptions. Instead, the user interacts with the algorithm as a black box that incorporates her feedback and provides new suggestions.

3.3.1 Layout. Our user interface (see Figure 2) follows a three-column table organized as follows:

- Each row corresponds to a user-created group;
- The left column in each group shows one "example," an item the user has already labeled;
- The middle column shows one "borderline," an instance that belongs to the group, but is very different from the examples; and
- The right column shows multiple "suggestions," instances that are representative of the user's group.

Our design builds directly on the findings of Forgarty et al. [23], which stated that, among several alternatives, interactive machine learning interfaces showing the best and worst instances were the most effective for participants. In our interface, "best" and "worst" correspond to suggestions and borderlines, and examples provide the user with a history of what she has labeled.

Every instance are allocated up to a 3cm square of screen space, allowing enough space to show images, text snippets, and data visualizations. The number of instances shown in the suggestions column is limited to five to ensure groups fit within the screen. Limiting what the user sees in the interface to this "tip of the iceberg" supports **VC2**.

However, before the user labels anything, we must address the "cold start problem," where model inferences cannot be made until the user provides sufficient initial information [45]. So, we overview the entire dataset with a small, diverse set of representative examples in an "unlabeled data" component. The representative examples in the unlabeled set are chosen using the same recommendation algorithm mentioned above, simply treating all remaining unlabeled data as a single group. This allows the user to choose a relevant starting point for her analysis, possibly ignoring irrelevant data. To support **VC1**, the unlabeled data list shows representative examples chosen from different clusters in the data.

3.3.2 Interactions. Instance are double-clicked to create a new group, causing that instance to be added as the first example for that group. When an instance is dragged into a group, that instance is labeled with the target group, and the instance is added to that group's list of examples. An arbitrary number of instances can be added as examples for each group. After these interactions, CHISSL predicts labels for all instances, and determines new borderlines and suggestions, possibly changing what instances are visible in the table. Below every instance is a "details button" that reveals more instances like the example above, supporting "details on demand." When clicked, a sidebar opens that the user can page through to see the additional instances, supporting **VC2**.

4 COMPUTATIONAL EVALUATION

We evaluate the classification accuracy of the CHISSL algorithm against a number of supervised and semi-supervised baselines across several diverse datasets. We seek to evaluate model performance as a function of the number of labels to explore the benefit of applying the CHISSL framework to reduce the labeling burden required of experts.

We evaluate the classification performance of CHISSL in two scenarios. First, where a user is trying to generate predicted class labels for a currently-existing, unlabeled data set, we employ transduction prediction. Second, where the user wants to generate a predictive classifier which can be applied in the future to previously unseen data instances, we leverage CHISSL for inductive prediction by generating semi-supervised predictions on the currently observed data set and then training an inductive, supervised classifier on this enhanced training set.

4.1 Data

CHISSL is evaluated on six data sets selected to explore a range of characteristics including the number of instances, the number of features, the number of classes, and the level of class imbalance.

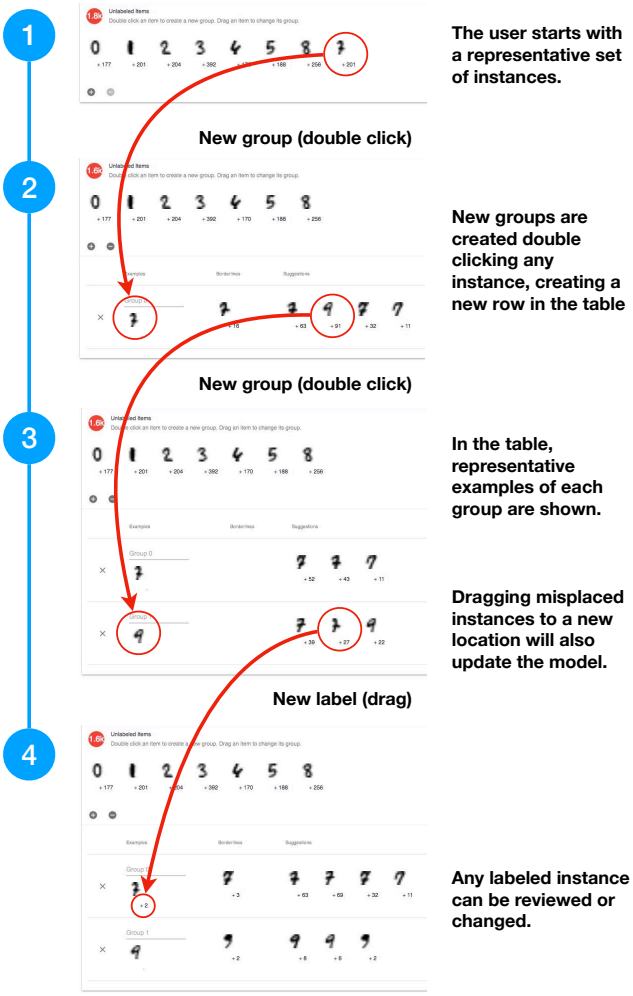


Figure 2: CHISSL Workflow—An illustration of several steps using CHISSL where the user starts without labeled instances (1) creates a new groups (2-3) and finishes after providing feedback the model (4). Red annotations indicate what the user selected and where that instance moved to.

Four data sets were obtained from the UCI Machine Learning Repository [20]. For the anuran dataset, the species, rather than the family or the genus, was used as the label. For the newsgroups a subset of categories were selected and grouped to use as labels. The key properties of each data set are summarized in Table 1. We perform feature reduction on each data set using non-negative matrix factorization (NMF) to reduce sparsity and correlations between features, and perform feature normalization by subtracting the mean and dividing by the standard deviation per feature.

4.2 Baseline Models

We compare the performance of CHISSL to a standard semi-supervised learning algorithm, Label Propagation (LP) [54], as well as a number of standard supervised algorithms including random forest (RF), AdaBoost (AB), logistic regression (LR), multi-layer perceptron (MLP), and naïve Bayes (NB) classifiers. Each baseline model is implemented using scikit-learn[1] and optimized using a random search strategy [7] to tune the hyper-parameters. Because we employ each baseline model for both the transduction task, where training is performed on a limited set of labeled data, and the induction task, where a large portion of the data is used for training, we separately optimize two versions of each classifier for performance on small or large numbers of available labels.

The full optimization proceeds in three steps. Firstly, the dimensionality of the NMF feature reduction step is optimized for classification performance. This parameter is allowed to vary simultaneously with the hyper-parameters for each classifier in a three-fold cross validation random search with over 100 hyper-parameter trials (the exact value depending on the dataset based on running time). The dimensionality reduction parameter is observed for each of the best performing models from each of the six classifiers, and the final value is selected by averaging these six values. Next, given this fixed NMF dimensionality, a second three-fold cross validation random search is employed on each full data set to select the final hyper-parameters of each classifier. These selected hyper-parameters are used for the inductive evaluation as they are optimized for large numbers of available training labels. Finally, we perform optimization of each model with a very limited number of labels, one per each class, with seven iterations of randomly selected labels. The parameters from this optimization were employed for the transductive evaluation where limited numbers of labels ($n_{\text{classes}} < n_{\text{labels}} < 100$) were available for training.

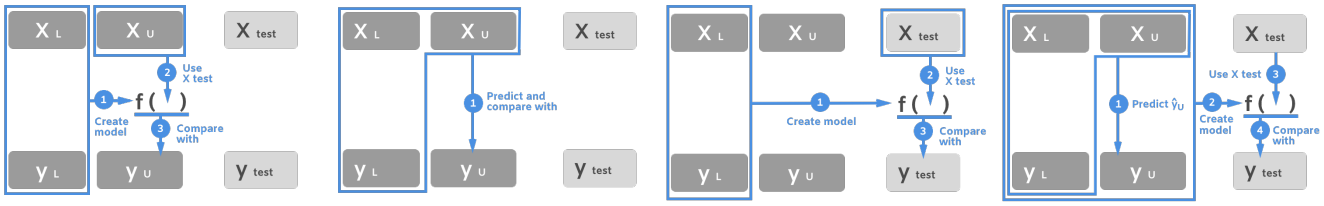
Table 1: Evaluation Data Sets and Benchmarking—We report the results of the benchmarking comparison between CHISSL and LP, indicating the time elapsed (seconds) for the CHISSL clustering step and the fit time for both CHISSL and LP different number of label assignments. We measure the speed advantage of CHISSL by taking the ratio of the LP fit time to the CHISSL fit time. The number of instances is n.

Data		Clustering (s)	n_{class} Labels - Fit (s)			100 Labels - Fit (s)		
Name	n	CHISSL	CHISSL	LP	Ratio	CHISSL	LP	Ratio
wine	178	0.0120	0.0009	0.0670	78.9	0.0009	0.0082	9.1
digits	5620	0.2259	0.0065	5.0603	783.1	0.0070	4.9980	716.2
anuran_species	7195	1.1879	0.0261	91.9489	3526.3	0.0280	91.6211	3271.0
human_activity	5620	7.4063	0.0253	74.5111	2942.2	0.0280	75.4146	2689.9
isolet	7797	8.5090	0.0352	123.4988	3504.0	0.0369	124.3182	3371.1
newsgroups	6513	1.4070	0.0239	0.8690	36.4	0.0236	0.5945	25.2

4.3 Evaluation Methodology

The evaluation process begins by partitioning the data into two disjoint sets—the training set $\mathbf{X}_{\text{train}}$ with 80% of the full data set, which is observed by the CHISSL and LP algorithms, and a held-out test set $\mathbf{X}_{\text{test}}$ with 20%, which is never observed by the semi-supervised learning algorithms. During the evaluation procedure, the training set is further divided into two components—a set of labeled training instances $\mathbf{X}_L$ with labels $\mathbf{y}_L$ and a set of unlabeled training instances $\mathbf{X}_U$. Initially, all data instances are considered to be unlabeled and new data instances are selected sequentially to be

[1] http://scikit-learn.org/stable/



(a) Induction (unlabeled) (b) Transduction (c) Induction (d) Bootstrapped induction

Figure 3: Evaluation Methodology—an illustration of the different approaches to enable comparison between SL and SSL approaches. (a) and (b) show how we evaluated off-the-shelf induction models against transduction learning models like CHISSL and Label Propagation (transduction evaluation task). (c) and (d) show how we evaluated off-the-shelf induction models against transduction models by bootstrapping the training set using transduction and then training on a larger dataset with an induction learning model (induction evaluation task).

labeled with the true class assignment $\mathbf{c}$ to mimic the process of an expert user assigning labels.

To initialize the process, one instance per class is selected randomly to be labeled, $\mathbf{y}_L \leftarrow \mathbf{c}_L$, and each successive instance thereafter is selected either randomly or using an AL procedure. For each data set, we select instances to label until there are 100 labeled instances. We stop the evaluation after 100 instances for practical reasons—the evaluation is computationally expensive to perform, and classification on large training sets is outside the intended use of our system. Furthermore, 100 instances represents a small labeling budget, equivalent to roughly five minutes of work with a similar task using a prototype version of the tool [3].

Each time an instance is labeled, we evaluate CHISSL and the baseline models in both the transductive and inductive context. In the transductive setting, we update or train each model using only the currently labeled data instances with $\mathbf{X} = \mathbf{X}_L$ and $\mathbf{y} = \mathbf{y}_L$, and perform prediction on the unlabeled training instances $\mathbf{X}_U$. The predicted classes for the unlabeled set are generated by each model using the transductive label assignment algorithm for CHISSL and LP and the predictive classification function, $f : \mathbf{R}^{\hat{m}} \rightarrow \{c_i\}$, for each of the supervised learning algorithms. We compare the model predictions $\hat{\mathbf{y}}_U^{\mathrm{m}}$ with the true class assignments, $\mathbf{c}_U$.

In the inductive setting, we leverage the predicted labels from each semi-supervised model, i.e., CHISSL (CH) and Label Propagation (LP), generated during the transduction step to train the supervised baseline models with $\mathbf{X} = \mathbf{X}_{\mathrm{train}}$ and $\mathbf{y} = \hat{\mathbf{y}}_{\mathrm{train}}^{\{\mathrm{CH,LP}\}}$ and use these trained classifiers to predict the labels on the held-out test data $\mathbf{X}_{\mathrm{test}}$. We evaluate the performance of each combination of models (e.g. CH and AB, LP and RF, etc.) by comparing $\hat{\mathbf{y}}_{\mathrm{test}}^{\mathrm{m}}$ and $\mathbf{c}_{\mathrm{test}}$. We compare the performance of each model when using the bootstrapped CHISSL labels with the performance when using the bootstrapped LP labels, as well as when using straightforward inductive learning on the labeled instances with $\mathbf{X} = \mathbf{X}_L$, and $\mathbf{y} = \mathbf{y}_L$. Fig. 3 summarizes these different evaluations.

In addition to exploring multiple evaluation data sets, we compare model performance across several other conditions. To select the next data instance to label, we compare the random selection of an instance with an Active Learning (AL) approach leveraging maximum entropy uncertainty sampling [36], where we use the relative tree distance between nearest labels of each class as a proxy for the class probability distribution. For each configuration of the data set, merge-tree weighting scheme, and next instance sampling method, we perform ten iterations of the evaluation procedure to measure the performance variability under different random selections of the initial instance to label for each class. We compare two different methods of selecting the edge-weights for the merge-tree, uniform weighting across all edges and the weighting heuristic described in the Appendix.

4.4 Results

We measure the classification performance using the macro average of the F1-score as a function of the number of labeled data instances. Fig. 4 illustrates the measured performance for the digits dataset for the transduction and induction learning tasks, respectively. For transduction, we compare CHISSL to each individual baseline model, while in the inductive case we perform a separate set of comparisons for each supervised baseline classifier. For example, in the left panel of Fig. 4, we compare the performance of the RF model when trained on $\mathbf{X}_{\mathrm{train}}$ using $\hat{\mathbf{y}}_{\mathrm{train}}^{\mathrm{CH}}$ as labels (blue) and $\hat{\mathbf{y}}_{\mathrm{train}}^{\mathrm{LP}}$ as labels (green) as well as when trained on $\mathbf{X}_L$ using $\mathbf{c}_L$ as labels (red). We perform a similar evaluation for each additional supervised baseline classifier type.

We compare overall performance of CHISSL with the baseline models using two metrics—the initial advantage of CHISSL and the total area between the F1-score versus number of labels curves (as shown in Fig. 4). The initial advantage captures the performance of CHISSL relative to the baseline model under a scenario with very limited available labels (one per class) while the area between the curves captures the longer-term relative performance between the two models in the range from n_{class} to 100 assigned labels.

We find that CHISSL typically outperforms most supervised baseline models while performing competitively with the semi-supervised LP baseline for most data sets. In particular, CHISSL achieved better performance than LP for the human activity and isolet data sets, both initially and across the full evaluation, and achieved better initial performance on the digits data set, while LP outperformed CHISSL consistently for the newsgroups data set. The results for the other data sets varied depending on the experimental conditions of the hierarchy weights and next label sampling method. We perform

Table 2: Initial Advantage—Wilcoxon signed rank test results comparing the initial F1-score (n_{class} labels) for CHISSL and LP for the transduction (T) and induction (I) modes (m) as well as the uniform (U) and centroid distance (CD) weighting (w) schemes. We report the z-value and the corresponding p-value in parentheses. Positive z-values (green) indicate that CHISSL outperforms LP while negative values (red) represent the opposite. * = 99.9% CI, ** = 99% CI, * = 95% CI.**

m	w	anuran	digits	human_activity	isolet	newsgroups	wine
T	U	-1.580 * (0.03666)	0.561 (0.28450)	1.172 (0.09260)	2.090 ** (0.00934)	-1.886 * (0.01660)	-1.478 * (0.04685)
	CD	0.764 (0.20262)	1.376 (0.05934)	2.090 ** (0.00934)	2.293 ** (0.00506)	-1.376 (0.05934)	-0.663 (0.24112)
I	U	-4.866 *** (<0.00001)	0.272 (0.62185)	3.062 ** (0.00103)	5.094 *** (<0.00001)	-5.558 *** (<0.00001)	-3.696 *** (0.00009)
	CD	0.545 (0.44391)	4.064 *** (0.00002)	4.756 *** (<0.00001)	6.316 *** (<0.00001)	-3.990 *** (0.00003)	0.180 (0.68826)

a set of Wilcoxon signed rank tests to compare the performance of CHISSL with LP across both the transduction and the induction task and the different experimental conditions. These results are shown in Table 2 and Table 3.

We also perform two Wilcoxon signed rank tests to compare the use of uniform versus centroid distance weighting and random sampling versus uncertainty sampling across the full set of results. We find that the centroid weighting scheme improves the initial advantage of CHISSL compared with LP relative to uniform sampling with $p < 0.001$ and a median improvement of 0.033, while there is no statistically significant difference for the total area between the curves for the two different weighting schemes. We find that uncertainty sampling improves the area between the curves of CHISSL relative to LP with $p = 0.038$ and a median improvement of 0.343, with no significant difference in the initial advantage. This is expected because sampling is applied to subsequent instance selections after the initial instance selection.

4.5 Benchmarking

In addition to comparing the classification accuracy of CHISSL, we perform a benchmarking comparison between CHISSL and LP on the full set of evaluation data sets. The results of this benchmarking analysis can be found in Table 1. We compare the fit time for the two algorithms, which is the time taken to update the model when given a set of new labeled instances L, for both n_{class} labels and for 100 labels. While LP is prohibitively slow for an interactive interface for data sets on the order of several thousand instances, CHISSL is tens to thousands of times faster, with transduction performed in under 100ms for all benchmark data. In a practical application, CHISSL and LP would not be tasked with assigning n labels at once, but rather with assigning one label at a time as it is received from an expert annotator. Because CHISSL is an incremental algorithm, the fit time for labeling n instances one-by-one is identical to that listed in Table 1. However, because LP is *not* incremental, it would be necessary to run the full update each time a new label is added In practice, the total required fit time would be on the order of n times that listed in Table 1, further increasing the advantage of CHISSL.

Table 3: Area Between Curves—Wilcoxon signed rank test results comparing the sum of F1-scores across all 100 labels for CHISSL and LP for the transduction (T) and induction (I) modes (m) as well as for the uniform (U) and centroid distance (CD) weighting (w) schemes and the random (R) and uncertainty sampling (UC) sampling (s) schemes. We report the z-value with corresponding p-value in parentheses. Positive z-values (green) indicate that CHISSL performs better than LP while negative values (red) represent the opposite. * = 99.9% CI, ** = 99% CI, * = 95% CI.**

m	w	s	anuran	digits	human_activity	isolet	newsgroups	wine
T	U	R	-1.376 (0.05934)	-1.478 * (0.04685)	1.580 * (0.03666)	1.070 (0.11413)	-2.293 ** (0.00506)	-2.191 ** (0.00691)
	U	UC	-1.478 * (0.04685)	1.784 * (0.02182)	2.191 ** (0.00691)	2.293 ** (0.00506)	-2.293 ** (0.00506)	-0.459 (0.33288)
	CD	R	0.663 (0.24112)	-0.357 (0.38627)	2.191 ** (0.00691)	2.293 ** (0.00506)	-2.090 ** (0.00934)	-1.988 * (0.01252)
	CD	UC	-0.866 (0.16881)	-2.191 ** (0.00691)	2.293 ** (0.00506)	2.293 ** (0.00506)	-2.293 ** (0.00506)	-0.255 (0.44459)
I	U	R	-4.763 *** (<0.00001)	-5.116 *** (<0.00001)	3.283 *** (0.00046)	1.399 (0.10533)	-6.235 *** (<0.00001)	0.169 (0.95890)
	U	UC	-4.623 *** (<0.00001)	5.028 *** (<0.00001)	6.360 *** (<0.00001)	5.742 *** (<0.00001)	-6.493 *** (<0.00001)	3.578 *** (0.00015)
	CD	R	0.029 (0.84821)	-1.413 (0.10220)	5.808 *** (<0.00001)	5.367 *** (<0.00001)	-4.837 *** (<0.00001)	-2.459 ** (0.00737)
	CD	UC	-2.429 ** (0.00804)	-5.742 *** (<0.00001)	6.405 *** (<0.00001)	5.403 *** (<0.00001)	-6.110 *** (<0.00001)	5.735 *** (<0.00001)

Our browser implementation of CHISSL performed transduction in under 2ms for our largest dataset with 45k instances.

5 USER STUDY

We conducted a user study to evaluate CHISSL's "tip of the iceberg" design against Active Learning (AL). For experimental control, the AL interface was integrated into CHISSL and used the same classification algorithm discussed above. However, in the AL version, the single most distant instance was presented to the user to label, and recommendations or borderlines were not visible.

5.1 Method

We conducted a within-subjects, in-laboratory user study (n = 25) to evaluate participants' annotation and transduction accuracy using three datasets with two treatments: CHISSL and Sequential, i.e., AL. Participants were a mix of undergraduates (n = 12) and graduate students (n = 13) at a large, public university. Gender was a mix of female (n = 14) and males (n = 11) as was major, which was split between computer/data science (n = 15) and social sciences (n = 10). Participants were provided either extra credit or a $5 gift card for their participation. Each session lasted approximately one hour. The study was approved by the university's IRB.

After completing a pre-questionnaire and a five minute training video, each participant completed two rounds of labeling three datasets and a post-questionnaire with each round differing by the interface (CHISSL or Sequential). Each labeling task was timed for five minutes. The three datasets were publicly available, pre-labeled datasets of images (ImageNet [18]), tabular numerical data (Boston Housing [24]), and text (20 Newsgroups[2]). Screenshots of the user interface for the CHISSL condition are shown in Fig. 5.

[2]http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

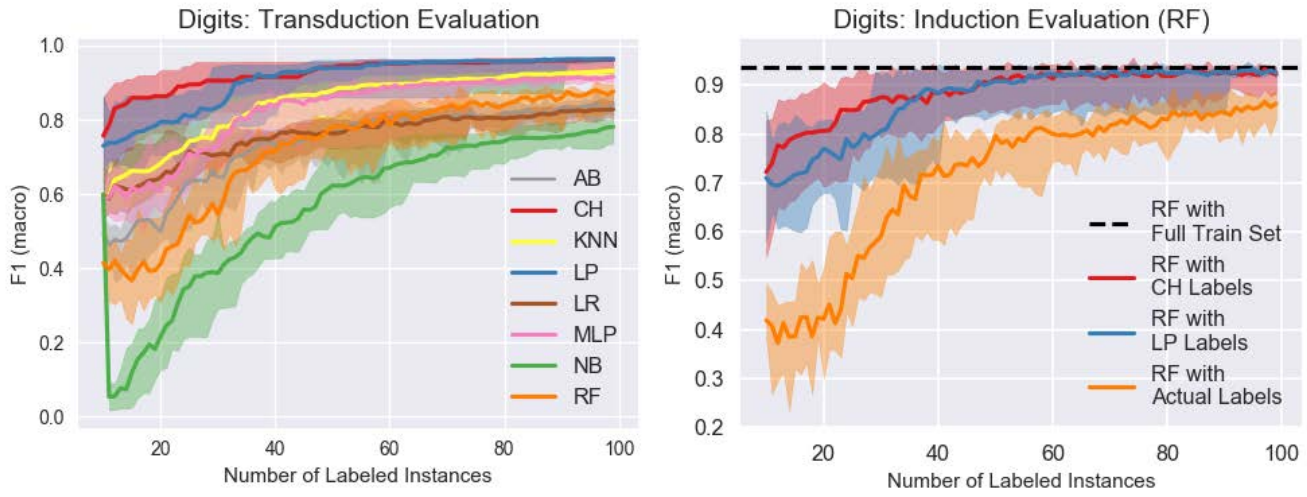


Figure 4: Classification Evaluation—(Left) Transduction Evaluation: Classification performance of CHISSL (CH, red) and the baseline comparison models for the digits data. The solid line for each classifier indicates the median performance across ten iterations of the evaluation procedure while the shaded band indicates the range of performance across these iterations. (Right) Induction Evaluation: Classification performance of RF models on the digits data using bootstrapped CHISSL (red) or LP (blue) labels. We compare this inductive application of the semi-supervised models with the use of the limited labeled set only as training data (orange) and the full set of true class labels on the training set (dashed black).

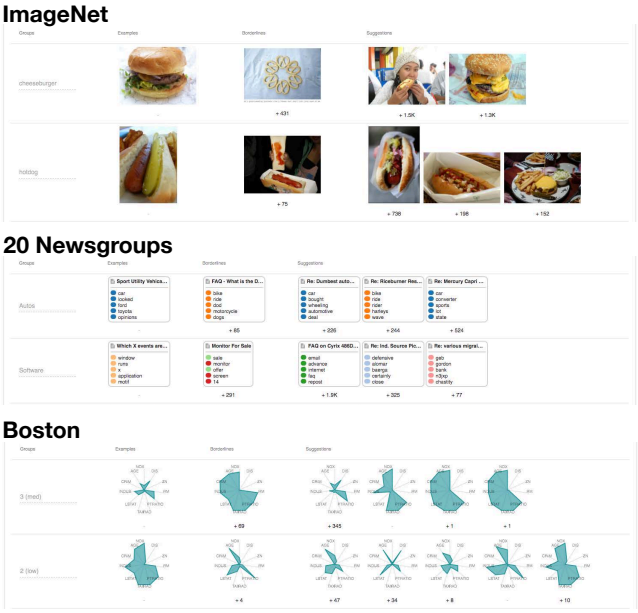


Figure 5: User study interface—screenshots of the first two rows of CHISSL for each of the datasets used in the study.

Datasets were manually down-selected to a few confusable classes (e.g., ice cream vs mashed potatoes images, software vs hardware discussion). For the Boston dataset we binned the housing price into one of six price levels $\{v.low = [0, 10),\ low = [10, 16),\ med = [16, 25),\ high = [25, 40),\ v.high = [40, 50),\ vv.high = 50\}$. While the ImageNet and Newsgroups datasets were fairly balanced between groups, the Boston dataset was imbalanced (e.g., mostly mid-level price). Because of the imbalance, we selected F1 (micro and macro) as the primary performance metric, and the number of instances labeled by the user serving as a secondary metric. We measured the accuracy of the users annotations as well as their transduction using both F1 measures.

To facilitate measurement of user accuracy and reduce variability, the interface was initialized with one pre-labeled instance from each class. These instances were manually chosen by the authors before the experiment, and were representative of their respective classes. The same pre-labeled instances were used for every participant across both conditions for each dataset. We also reduced variability by removing features from CHISSL that were unnecessary for the study including the unlabeled data component and the ability to create, delete, or rename groups.

To control for learning effects, participants were randomly assigned to one of six treatments which counterbalanced two factors: the order of the interface (i.e., each round) and the datasets presented each round. After completing the first round of tasks for a given interface, the participant would then move on to the other interface for the same datasets and a second post-questionnaire. The order that the datasets were provided was randomly permuted into one of three orders, and this order was repeated for both interface rounds.

5.2 Results

In general, we found that user labeling performance (F1 micro/macro scores) was better for CHISSL than Sequential for the ImageNet and Boston datasets while the opposite occurred for the Newsgroups dataset. Figure 4 provides the Wilcoxon Signed-Rank tests including the z score and p-value. Figure 6 shows kernel density plots of F1 micro and macro measures per dataset.

Table 4: User performance—Wilcoxon Signed-Rank tests for each dataset. Z-score are provided with the corresponding p-value. Positive Z-scores (green) indicate CHISSL performed better than Sequential; negative values (red) represent the opposite. * = 99.9% CI, ** = 99% CI, * = 95% CI.**

	F1 Measure	Boston	Newsgroups	ImageNet
Annotation	Macro	0.0968 *** (0.00004)	0.0246 (0.3525)	-0.0066 (0.4261)
	Micro	0.0957 *** (0.00007)	0.0468 (0.1065)	-0.0008 (0.8949)
Transduction	Macro	-0.0930 *** (0.00002)	-0.0772 *** (0.000002)	0.01581 ** (0.0342)
	Micro	0.6641** (0.0173)	-0.0753 *** (0.000003)	0.0168 *** (0.0088)
Number of labels		-6.0000 (0.3001)	5.5 (0.1872)	-0.4249 (0.9886)

From these figures, we can make three observations. First, from Tab. 4, we find that CHISSL lead to more accurate labels with over 99% confidence in general on the Boston dataset as well as the Transduction-based labels for ImageNet. Alternatively, we find that Sequential performed better for the Transduction labels in Newsgroups as well as the F1-Macro for Transduction Boston dataset. Both of these differences are illustrated in Fig. 6 where each plot shows probability density functions for the user-level F1 measures for each dataset and interface. Vertical lines indicate the average score for that respective dataset-interface pair. Situations where the statistical test indicate significant differences between the interfaces have vertical lines that are clearly separable.

A second observation is that users performed markedly different in their labeling task for each dataset. We found that that participants performed best on ImageNet (e.g., F1-Macro for User Labels was M = 0.932, SD = 0.034 for CHISSL and M = 0.934, SD = 0.059 for Sequential) while participants performed worst on the Boston dataset (e.g., F1-Macro for User Labels was M = 0.407, SD = 0.091 for CHISSL and M = 0.315, SD = 0.044 for Sequential). This result was expected as the items in the ImageNet dataset were much easier to understand than the multiple dimensions demonstrated in the radar plots for the Boston dataset. In addition, this result is consistent with participants' post-questionnaire in which 23 out of the 25 participants reported that the ImageNet was the easiest dataset for either CHISSL or Sequential.

Last, we find that each interface did not have a significant effect on the number of labels for either dataset. This result was a secondary check to understand if there were a trade-off between each of the interface in the scalability of the labels. However, we did find that the number of labels differed per dataset. For example, in the Newsgroups dataset, users labeled many fewer records (CHISSL was M = 51.48, SD = 29.586 while Sequential was M = 44.24, SD = 15.344) than the ImageNet dataset (CHISSL was M = 102.16, SD = 52.196 and Sequential was M = 100.20, SD = 30.396).

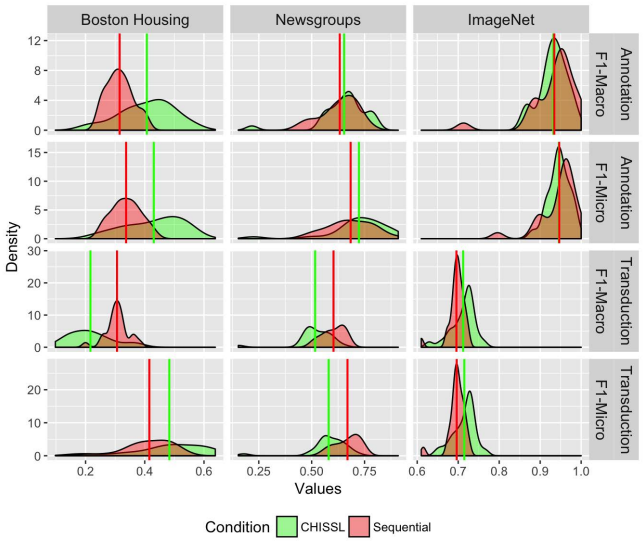


Figure 6: Density Plots—Performance measures (F1 Macro/Micro) scores kernel density plots for CHISSL and Sequential by Dataset (column) and Measure (row). Vertical lines are mean values for each distribution. Kernel density use Gaussian smoothing kernel with adjustment factor equal to one.

To assess possible effects of learning or user-level attributes, we considered multivariate linear regression to explain each performance metric using a variety of independent variables (see supplemental materials). We considered eleven additional independent variables including major (Computing or Social Science), student level (Graduate or Undergraduate), gender, ordering (dataset or interface), as well as pre-questionnaire self-assessment questions (e.g., familiarity with machine learning, text analysis, radar plots). We found little evidence that any of these additional variables had a significant effect on performance. One interpretation points to the general use of the interfaces in which major, student level, or familiarity with machine learning did not provide an advantage (or disadvantage) to users. This suggests that CHISSL can be used by a general population rather than specific sub-groups. We found some evidence of a negative effect of the number of labels and performance for the Boston dataset. We suspect that given the difficulty of that dataset, users who rushed were made more misclassifications.

We also considered user feedback in the post-questionnaire to assess any possible issues with either interface. For interface and study feedback, users rated nine questions on a 7-level Likert Scale (7 = Strongly Agree, 1 = Strongly Disagree). Overall, users found the task interesting (M = 5.12, SD = 1.624), the training as sufficient (M = 5.76, SD = 1.153), as well as that the interfaces were easy-to-use (M = 5.64, SD = 1.290), fast and quick (M = 6.10, SD = 1.233), and well-integrated (M = 5.44, SD = 1.163). T-tests did not show a significant effect of the interface on the user responses.

Last, 15 of the 50 questionnaires included a user response in the open-ended feedback. Users' comments varied across several topics. For example, some users noted that the tasks were very

different for each dataset: "Time to review the emails/text was longer. It was not necessarily harder just took longer. The Boston dataset was relatively easy". Others suggested additional features like an "undo" button: "Undo option would be great, because I did mistakes which I wanted to reverse". However, several commented on their general enjoyment of the study: "Loved this study!! I would definitely participate in a similar study like this one!"

6 DISCUSSION

6.1 Interpretation of Results

6.1.1 Computational Evaluation. CHISSL usually performed better than supervised learning algorithms across the evaluation datasets with few labels. This is consistent with the main assumption of semi-supervised learning, that unlabeled instances are beneficial. In some cases, CHISSL matched or outperformed the baseline semi-supervised learning algorithm, Label Propagation [54]. However, CHISSL's $O(n)$ time complexity facilitates classification at significantly faster rates than the baseline. Due to the efficiency of the classification algorithm, users may be able to provide more labels to achieve accuracy comparable to more accurate but slower models in the same amount of time.

The benefit of our centroid distance weighting scheme over uniform weighting was clear, but sometimes CHISSL performed better under random sampling compared to uncertainty sampling. Either CHISSL was more robust to a sub-optimal sampling technique, or Active Learning (AL) benefitting more from uncertainty sampling. Alternatively, uncertainty sampling could be biased to select outliers, which are less helpful to improving the overall classification. While our user interface shows "borderlines" selected with uncertainty sampling, it also shows more representative instances in the "suggestions" column, allowing the user to better handle these cases.

6.1.2 User Study. Compared to a traditional sequential AL interface, our design helped study participants produce more accurate transductions for the ImageNet and Boston Housing tasks. Additionally, users' annotations of the data were more accurate for the Boston Housing conditions. It appears that giving participants a choice over what to label allowed them to select more beneficial instances for the model and to avoid misclassifications. Sequential AL interfaces may bias users towards more annotation errors by forcing the user to label instances they are uncertain about. However, CHISSL was clearly less helpful for participants for the 20 Newsgroups conditions. In this case, less pre-attentive cues [51] were available to facilitate a rapid visual search compared to the other two conditions. Improving the upstream analytics pipeline and including document summarization with keyword highlighting in the visualization would likely improve CHISSL for this use case.

6.2 Other Considerations for Clustering

The effectiveness of CHISSL is strongly dependent on hierarchical clustering. For our study, we used Ward clustering [50], though any hierarchical clustering algorithm could be used. In practice, we observed that Ward clustering produced better results than other techniques like minimum or maximum linkage. "Bad clusters," either due to poor feature engineering or cluster parameters, tended to manifest in the interface as a single giant group with a few smaller groups. Dragging instances out of the giant group tended to have little overall effect on the transduction. In general, clustering algorithms are sensitive to their parameters and random seeds, so two different clusterings of the same data could produce different transductions in CHISSL. Ensemble approaches [43] could use this effect by blending the separate transductions into a single, possibly more accurate one.

The hierarchical clustering algorithm also affects what instances are shown to the user. We used a heuristic to determine which instances to display in the "Suggestions" column by selecting the deepest instance within each sub-cluster. The intuition was that the deepest instance would be fairly representative of that sub-cluster, and different from other representatives in other sub-clusters. In an earlier version of CHISSL we displayed the least distant instance from its group in each sub-cluster. However, this was problematic, as it often hid interesting variability within the cluster.

Edge weights in the cluster hierarchy are free parameters that can be tuned to improve classification performance. We used a weighting based on the Euclidean distance between cluster centroids, but other functions such as a distribution distance, e.g., K-L divergence [32], could be used instead. It was not immediately clear what benefit more complicated weighting schemes add to the overall classification accuracy of the system, so we opted for a fast and simple approach and leave this question for future work.

6.3 Limitations

Some application domains like object recognition in images have orders of magnitudes more classes and instances. Our classification algorithm scales independently from the number of classes, and supports an arbitrary number of labels per class. Labeling one new instance is $O(n)$ regardless of the number of classes or labeled instances. However, our evaluation of CHISSL did not consider a scenario where a large number of instances are labeled, as this is outside the intended use case of the tool. We assume that CHISSL is less accurate than state-of the-art supervised classification algorithms in these cases.

Our experience is that CHISSL becomes frustrating to use when some groups are not visible on the screen, requiring the user to scroll and drag simultaneously. We also noticed that the tool sometimes hides interesting sub-clusters or outliers, which could be detrimental to users' trust in the system. In practice, CHISSL's current design is most effective with ten or fewer classes—making the interface more compact would increase the number of classes visible at the expense of showing fewer representative examples. However, the "tip of the iceberg" design alone likely will not scale to thousands of classes, revealing an opportunity for future work, perhaps by extending CHISSL to support hierarchical classification.

A critical limitation of our approach stems from the assumption that the classification and clustering are consistent with each other. In other words, we expect the decision boundary to lie in a less dense region of the feature space, and therefore be detectable by the clustering algorithm. When this assumption does not hold, we expect CHISSL to perform poorly compared to traditional supervised techniques, e.g., support vector machines. Also, if the representation matrix has redundant or useless columns (features) for the

classification task, our approach would not perform as well as techniques that incorporate feature selection. A solution would be to employ CHISSL iteratively, using the user's labels or transduction to reduce the dimensinoality of the representation matrix, which would improve the subsequent clustering. If this produces a better transduction, the process can be repeated.

6.4 Lessons Learned and Design Improvements

Change blindness was an issue that we discovered from the user study that we had not anticipated. Some users noted that the interface was changing "too fast," which we interpret to mean they were sometimes unable to track all effects caused by labeling an instance. Following the study, we addressed this by indicating which instances have joined or left each group after the user provides feedback. This clarifies what effect each action has on the transduction. These signals could help the user decide when the model has converged.

We have also integrated a dimension reduction plot into the current version of CHISSL. These plots are frequently used by Visual Interactive Labeling tools, and help to summarize the feature space with a spatial overview. This helps to identify outliers or groups that might need to be split.

7 CONCLUSION

For label elicitation, we found that CHISSL is

- Rapid—classification without representation allows the interface to quickly provide user feedback after each interaction within a few miliseconds;
- Accurate—our algorithm is competitive with off the shelf supervised and semi-supervised classifiers with small amounts of labeled data; and
- Helpful—when visual search is supported, users' transductions were more accurate for structured and image data compared to active learning.

Besides addressing limitations already discussed, our future work will focus on the bigger picture and additional applications. Our next goals are to enable an analyst to quickly query data from a stream, organize and annotate that data with CHISSL, build an inductive model, enrich/classify streaming data, and then monitor the stream for trends and anomalies using the analysts own model.

We plan to apply CHISSL to real world data and application domains including disaster monitoring, insider threat detection, and geospatial analysis. We hypothesize that because CHISSL greatly reduces the effort to build machine learning models, these models can be created and refined at pace with such highly variable streaming data.

ACKNOWLEDGMENTS

The research described in this paper was conducted under the Laboratory Directed Research and Development Program at Pacific Northwest National Laboratory, a multi-program national laboratory operated by Battelle for the U.S. Department of Energy.

REFERENCES

[1] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120.

[2] Saleema Amershi, James Fogarty, and Daniel Weld. 2012. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 21–30.

[3] Dustin Arendt, Caner Komurlu, and Leslie M Blaha. 2017. CHISSL: A Human-Machine Collaboration Space for Unsupervised Learning. In *International Conference on Augmented Cognition*. Springer, 429–448.

[4] Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. 2014. Local algorithms for interactive clustering. (2014).

[5] Maria-Florina Balcan and Avrim Blum. 2008. Clustering with interactive feedback. In *ALT*. Springer, 316–328.

[6] Sugato Basu, Ian Davidson, and Kiri Wagstaff. 2008. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

[7] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.* 13 (Feb. 2012), 281–305. http://dl.acm.org/citation.cfm?id=2188385.2188395

[8] Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Fellner, and Michael Sedlmair. 2018. Comparing Visual-Interactive Labeling with Active Learning: An Experimental Study. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 298–308.

[9] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. 2017. A Unified Process for Visual-Interactive Labeling. (2017).

[10] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 11.

[11] Avrim Blum and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. (2001).

[12] Lydia Boudjeloud-Assala, Philippe Pinheiro, Alexandre Blansché, Thomas Tamisier, and Benoît Otjacques. 2016. Interactive and iterative visual clustering. *Information Visualization* 15, 3 (2016), 181–197.

[13] Eli T Brown, Jingjing Liu, Carla E Brodley, and Remco Chang. 2012. Dis-function: Learning distance functions interactively. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*. IEEE, 83–92.

[14] Pierrick Bruneau, Philippe Pinheiro, Bertjan Broeksema, and Benoît Otjacques. 2015. Cluster sculptor, an interactive visual clustering system. *Neurocomputing* 150 (2015), 627–644.

[15] Keke Chen and Ling Liu. 2004. Clustermap: Labeling clusters in large datasets via visualization. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 285–293.

[16] Keke Chen and Ling Liu. 2004. VISTA: Validating and refining clusters via visualization. *Information Visualization* 3, 4 (2004), 257–270.

[17] Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. 2012. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 443–452.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[19] Marie Desjardins, James MacGlashan, and Julia Ferraioli. 2007. Interactive visual clustering. In *Proceedings of the 12th international conference on Intelligent user interfaces*. ACM, 361–364.

[20] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[21] Niklas Elmqvist and Jean-Daniel Fekete. 2010. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (2010), 439–454.

[22] Alex Endert, Patrick Fiaux, and Chris North. 2012. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 473–482.

[23] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: interactive concept learning in image search. In *Proceedings of the sigchi conference on human factors in computing systems*. ACM, 29–38.

[24] David Harrison Jr and Daniel L Rubinfeld. 1978. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management* 5, 1 (1978), 81–102.

[25] Florian Heimerl, Steffen Koch, Harald Bosch, and Thomas Ertl. 2012. Visual classifier training for text document retrieval. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2839–2848.

[26] Benjamin Höferlin, Rudolf Netzel, Markus Höferlin, Daniel Weiskopf, and Gunther Heidemann. 2012. Inter-active learning of ad-hoc classifiers for video visual analytics. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*. IEEE, 23–32.

[27] Dong Hyun Jeong, Caroline Ziemkiewicz, Brian Fisher, William Ribarsky, and Remco Chang. 2009. iPCA: An Interactive System for PCA-based Visual Analytics. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 767–774.

[28] Ian T Jolliffe. 1986. Principal component analysis and factor analysis. In *Principal component analysis*. Springer, 115–128.

[29] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).
[30] Joseph B Kruskal. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29, 1 (1964), 1–27.
[31] 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*. ACM, 3075–3084.
[32] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[33] Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F Stewart, and Adam Perer. 2018. Clustervision: Visual Supervision of Unsupervised Clustering. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 142–151.
[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[35] Hanseung Lee, Jaeyeon Kihm, Jaegul Choo, John Stasko, and Haesun Park. 2012. iVisClustering: An interactive visual document clustering via topic modeling. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1155–1164.
[36] David D. Lewis and Jason Catlett. 1994. Heterogeneous Uncertainty Sampling for Supervised Learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 148–156.
[37] Zhicheng Liu and Jeffrey Heer. 2014. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization & Computer Graphics* 1 (2014), 1–1.
[38] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.
[39] J Möhrmann, S Bernstein, T Schlegel, G Werner, and G Heidemann. 2011. Improving the usability of interfaces for the interactive semiautomatic labeling of large image data sets. *Human Computer Interaction. Design and Development Approaches* (2011), 618–627.
[40] Daniel Müllner et al. 2013. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software* 53, 9 (2013), 1–18.
[41] Jose Gustavo S Paiva, William Robson Schwartz, Helio Pedrini, and Rosane Minghim. 2015. An approach to supporting incremental visual data classification. *IEEE transactions on visualization and computer graphics* 21, 1 (2015), 4–17.
[42] Meg Pirrung, Nathan Hilliard, Nancy O'Brien, Artem Yankov, Nathan O Hodas, et al. 2018. SHARKZOR: Human in the Loop ML for User-Defined Image Classification. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*. ACM, 29.
[43] Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review* 33, 1-2 (2010), 1–39.
[44] Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*.
[45] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 253–260.
[46] Christin Seifert and Michael Granitzer. 2010. User-based active learning. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*. IEEE, 418–425.
[47] Jinwook Seo and Ben Shneiderman. 2002. Interactively exploring hierarchical clustering results. *Computer* 35, 7 (2002), 80–86.
[48] Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.
[49] Akash Srivastava, James Zou, and Charles Sutton. 2016. Clustering with a reject option: Interactive clustering as bayesian prior elicitation. In *KDD 2016 Workshop on Interactive Data Exploration and Analytics*.
[50] Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58, 301 (1963), 236–244.
[51] Colin Ware. 2012. *Information visualization: perception for design*. Elsevier.
[52] Leland Wilkinson. 2018. Visualizing Big Data Outliers through Distributed Aggregation. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 256–266.
[53] Xiaojin Zhu. 2005. *Semi-Supervised Learning Literature Survey*. Technical Report 1530. Computer Sciences, University of Wisconsin-Madison.
[54] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. (2002).