AnchorViz: Facilitating Semantic Data Exploration and Concept Discovery for Interactive Machine Learning

JINA SUH, SOROUSH GHORASHI, and GONZALO RAMOS, Microsoft Research, Redmond, Washington
NAN-CHEN CHEN, University of Washington, Seattle, Washington
STEVEN DRUCKER, JOHAN VERWEY, and PATRICE SIMARD, Microsoft Research, Redmond, Washington

When building a classifier in interactive machine learning (iML), human knowledge about the target class can be a powerful reference to make the classifier robust to unseen items. The main challenge lies in finding unlabeled items that can either help discover or refine concepts for which the current classifier has no corresponding features (i.e., it has *feature blindness*). Yet it is unrealistic to ask humans to come up with an exhaustive list of items, especially for rare concepts that are hard to recall. This article presents *AnchorViz*, an interactive visualization that facilitates the discovery of prediction errors and previously unseen concepts through human-driven semantic data exploration. By creating example-based or dictionary-based anchors representing concepts, users create a topology that (a) spreads data based on their similarity to the concepts and (b) surfaces the prediction and label inconsistencies between data points that are semantically related. Once such inconsistencies and errors are discovered, users can encode the new information as labels or features and interact with the retrained classifier to validate their actions in an iterative loop. We evaluated AnchorViz through two user studies. Our results show that AnchorViz helps users discover more prediction errors than stratified random and uncertainty sampling methods. Furthermore, during the beginning stages of a training task, an iML tool with AnchorViz can help users build classifiers comparable to the ones built with the same tool with uncertainty sampling and keyword search, but with fewer labels and more generalizable features. We discuss exploration strategies observed during the two studies and how AnchorViz supports discovering, labeling, and refining of concepts through a sensemaking loop.

CCS Concepts: • **Human-centered computing → Interactive systems and tools**;

Additional Key Words and Phrases: Interactive machine learning, visualization, error discovery, semantic data exploration, unlabeled data, concept discovery, machine teaching

ACM Reference format:
Jina Suh, Soroush Ghorashi, Gonzalo Ramos, Nan-Chen Chen, Steven Drucker, Johan Verwey, and Patrice Simard. 2019. AnchorViz: Facilitating Semantic Data Exploration and Concept Discovery for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 10, 1, Article 7 (August 2019), 38 pages.
https://doi.org/10.1145/3241379

The reviewing of this article was managed by special issue associate editors Mark Billinghurst, Margaret Burnett, and Aaron Quigley.

Authors' addresses: J. Suh, S. Ghorashi, G. Ramos, S. Drucker, J. Verwey, and P. Simard, Microsoft Research, 1 Microsoft Way, Redmond, WA, 98052; emails: {jinsuh, sorgh, goramos, sdrucker, joverwey, patrice}@microsoft.com; N.-C. Chen, University Washington, Seattle, WA, 98195; email: nanchen@uw.edu.

2160-6455/2019/08-ART7 $15.00
https://doi.org/10.1145/3241379

1 INTRODUCTION

Interactive machine learning (iML) is a growing and active field in machine learning (ML) that emphasizes building models with humans in the loop [17]. A typical ML process involves (1) selecting unlabeled items (hereafter, we refer to a data point or an example that can be labeled as a whole as an "item") that the model should understand and learn from, (2) examining and labeling the items appropriately according to the desired behavior of the model, and (3) choosing the right representations (i.e., features) that describe the salient aspects of the labeled items that correctly differentiate the target concept from the rest. In the traditional ML workflow, these steps are performed sequentially and independently in batches; training and tuning of these models take significant amount of time such that it is impractical for humans (i.e., model developers) to interact with the model in real time. On the other hand, an iML workflow allows humans to take advantage of interactive training to efficiently correct and teach the model and to focus on iteratively providing the most appropriate input given the model's current deficiencies; as these models constantly get feedback from humans, their evolving directions can better align with the developers' goals.

Interactive training means that any model-changing input from humans (e.g., a new label, a new feature) triggers a real-time training of the model such that any subsequent interaction reflects the newly trained model's updated behavior. Given the latest state of the model, an iML system may focus on helping humans find [5] or generate [20] the most appropriate item to label (interactive sampling), provide appropriate labels to correct the model's current predictions [14, 18, 20, 47] (interactive labeling), or ideate [8] or select [55] the most appropriate feature to describe the labeled data (interactive featuring). We illustrate this conceptual workflow of an iML system in Figure 1. Our work focuses on the interactive sampling step to efficiently explore (or sample) and label unlabeled items that the current model will predict incorrectly (i.e., *errors in unlabeled items*) because, if the size of the unlabeled dataset is large, it is undesirable or impractical for a person to label the whole dataset in an iML workflow.

One common, algorithmic approach for selecting unlabeled items is sampling based on uncertainty (e.g., items with prediction scores near a decision boundary) or uniform distribution (i.e., items uniformly sampled based on the prediction score distribution). Although these sampling methods can help pull out prediction errors, often the selected items can be ambiguous (i.e., hard to judge) or random (i.e., the probability of finding a prediction error is as good as random selection). Errors with high prediction confidence, which can be disastrous in high-risk scenarios, are not likely to be selected by these sampling algorithms. Hence, humans may not even be aware of such defects in their models. Furthermore, these sampling methods do not fully leverage the value of human insight; instead, these methods treat humans as mindless label generators, and focusing on inherently ambiguous patterns cause them to feel annoyed, lose track of teaching progress, or lose interest [3]. As the discovered prediction errors are mostly uncertain, it can be challenging for humans to make sense of the errors to provide useful inputs.

Interactive sampling, on the other hand, allows humans to take advantage of their knowledge about the target model and their understanding of the current model to explore and identify unlabeled items that the model should learn from. Specifically, in classification problems, humans usually possess knowledge about the target class; they can come up with hypotheses on what underlying *concepts* (i.e., abstract notions that altogether represent the target class) are missing from the model or are difficult for the model to interpret and try to fix the errors by providing more labeled items or adding features. Such items could be found prescriptively (e.g., text search) if humans could enumerate the missing information. In our work, we leverage semantic models (i.e., models that have semantically meaningful feature representations [25]) to explore the dataset when the missing information is not readily available or known until revealed. We propose an interactive visualization, *AnchorViz* [11], to facilitate this interactive sampling process. We focus on

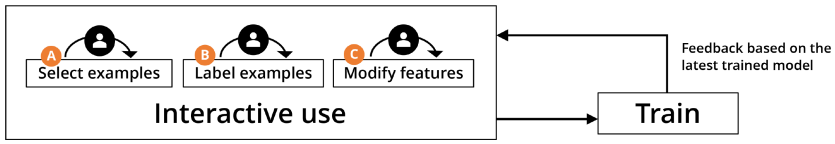

Fig. 1. Interactive machine learning workflow. An iML workflow involves a quick, iterative loop between an interactive use of the iML system and interactive training of the target model. Humans can interactively select examples (A), label examples (B), or modify features (C) such that specific requirements of the target model are met or identified deficiencies of the latest trained model are corrected. After such human inputs, the model is quickly retrained for the human in the loop to interact with the model, assess its quality, and determine the next best action to take.

helping users to *sift* the dataset with magnets that represent semantic concepts that we call *anchors* and to leverage their domain knowledge in finding unlabeled items to which the model is blind.

We evaluated AnchorViz in two user studies. The first user study examined the users' exploration strategies and the use of anchors in improving an in-progress binary classifier (i.e., the model developer spent some amount of effort already) and focused on evaluating the interactive sampling step (Figure 1(A)) for a fixed model (i.e., interactive training and featuring is disabled in an iML loop). Our results show that items highlighted through anchors are more likely to be prediction errors for which the current classifier has no corresponding features (hereafter referred to as "feature blindness errors") [44]. In addition, the participants used a variety of strategies to explore the dataset and distill positive and negative concepts of the target class by creating anchors.

The second user study was a controlled experiment that compared the use of AnchorViz (a human-driven approach) to the use of uncertainty sampling (a machine-driven approach) for building a classifier from scratch in an end-to-end iML loop (Figure 1). Our results show that AnchorViz helped users to create a comparable model (i.e., similar in test accuracy) with fewer labels and more generalizable features than the use of an uncertainty sampling and keyword search interface. Achieving a certain model performance with fewer labels is beneficial for reducing the human labeling cost [56]. In addition, most participants preferred AnchorViz over the uncertainty sampling and keyword search method, while a few preferred having access to both.

The contributions of this work are as follows: First, we highlight the opportunity for leveraging semantic data exploration to locate classifier errors in unlabeled items. Second, we design, build, and present AnchorViz to support such exploration through an interactive visualization. Third, we evaluate AnchorViz through two user studies and discuss people's exploration strategies and AnchorViz's support for discovering, labeling, and refining concepts. As we discuss in the next section, our work differentiates from existing literature in iML. Our work studies an end-to-end experience of building a model where neither feature nor label sets are fixed. Our results show that AnchorViz has the potential to improve the iML experience by requiring fewer labels and producing more meaningful features, while opening a new research space for semantic interactions within an iML system.

2 MOTIVATING SCENARIOS

Before we describe the background and details of AnchorViz, we provide the following two scenarios to illustrate how one can create and use concept magnets to explore unlabeled items while building classification models. In both scenarios, the model builder is unaware of an important concept such that a training example that includes the concept or a feature that describes the concept cannot be produced *a priori* or recalled in any prescriptive manner; such a concept, when discovered and added to the training set, would improve the generalization of the model. We base our scenarios and the design and evaluation of AnchorViz on building binary classifiers for a text dataset.

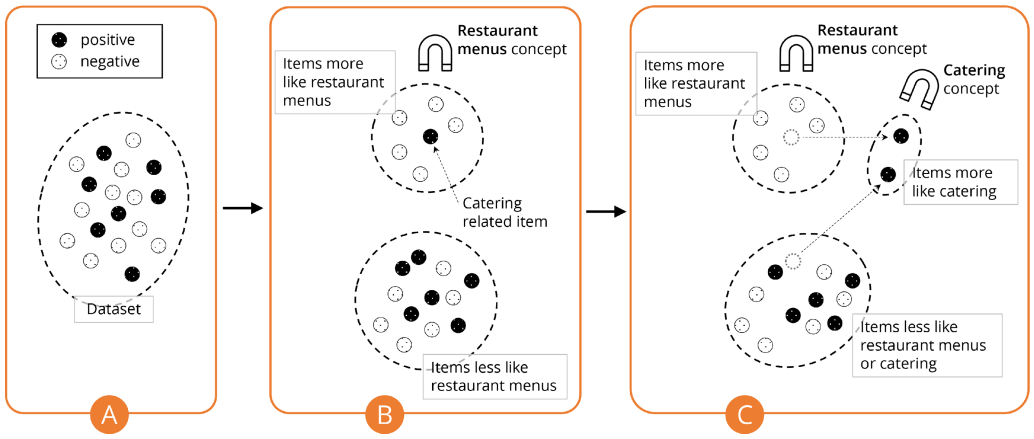

Fig. 2. The use of concept magnets for separating the dataset when building a recipes classifier. A dataset contains a mix of predicted positive or negative items according to the current classification model (A). When a *restaurant menu* concept magnet is brought to a dataset, items that are more like the restaurant menu concept are pulled toward the magnet while items that are less like the restaurant menu concept remain (B). Notice a predicted positive catering item is amid restaurant menu related items. A *catering* concept magnet is added to pull additional items that are catering related so that such items can be inspected and corrected (C).

2.1 Discovering a Catering Concept for a Recipes Classifier

A food blogger wants to build a binary classifier for finding recipe webpages. We illustrate the use of concept magnets in this scenario in Figure 2. The writer already considers restaurant menu pages as negative, so he wants to separate out restaurant menu webpages from the rest of the dataset to make sure they are predicted as negative (2(A)). He uses existing restaurant menu pages to create a *restaurant menu* concept magnet to pull webpages like the concept of restaurant menus (2(B)). Among the similar items that are pulled towards the *restaurant menu* concept magnet, the writer notices some items that are being predicted as positive, which is unexpected. Upon inspection of those predicted positive items, he discovers that they are catering webpages. The writer considers catering webpages as not belonging to recipes, labels the discovered catering webpages as negative, and creates a *catering* concept magnet. With both *restaurant menu* and *catering* concept magnets, he can now see if there are other webpages pulled towards both magnets that are incorrectly predicted as positive (2(C)). He adds a feature to describe the *catering* concept, and a new model is trained to reflect that the webpages pulled towards the *catering* concept magnet are correctly classified as negative.

2.2 Exploring an Unfamiliar Dataset while Building a Classifier from Scratch

A student is doing research on online communities that discuss issues related to current events. She is interested in conversations about gun issues and wants to build a classifier to separate posts about guns from the rest. She is unfamiliar with the dataset and what it contains but knows that the content she is looking for is there. She starts her task by creating concept magnets about the *firearms* and *gun actions* concepts, which starts attracting items that contain terms used to describe the concepts (e.g., shoot, pistol, rifle). From this set of items, the student selects items which she expects are positive examples for the main concept, so she can label them. To her surprise, some of these items are not positive examples; some items are about the *photography* concept and contain the term “to shoot” or “shot”. The student has learned an unexpected aspect about the dataset

she did not know before, the presence of photography items. This knowledge helps her to create a photography concept magnet that separates photography items from gun ones that she labels as negative examples. She also adds the *photography* concept as a feature to describe the negative examples she added to the training set. As the student repeats the process described above, she not only keeps improving her *gun issues* classifier, she progressively gains familiarity with the dataset and the information contained within.

3 BACKGROUND AND RELATED WORK

In an iML setting, humans build models through iteratively training the model and providing training items and features based on the current model's state in a quick and continuous loop [3]. This model-building approach has shown its success in building a well-performing model using fewer features [68]. In addition, it provides meaningful interactions to improve the user's trust and understanding of the system [62]. This interaction between humans and machines goes beyond simply treating humans as label oracles and requires thoughtful design and careful user studies [3]. Like traditional ML, iML systems need to support finding items and features to which the current model is blind for better generalization performance in the real world.

As illustrated in Figure 1, an iML workflow involves selecting items to label, labeling items, and featuring in an iterative loop, where one or more of these steps could incorporate human inputs (i.e., interactive feature ideation) or leverage automatic or algorithmic techniques (i.e., uncertainty sampling). Existing literature on iML systems focus on soliciting human labels while holding features constant or allowing the algorithm to select features [14, 17, 18, 20] or soliciting human features while holding labels constant [8, 12, 25]; very few works have tried to solicit both human labels and features [5]. Others leverage a dual learning paradigm which allows users to label features [55, 67] while holding features constant. We focus our work on facilitating human-driven, interactive selection of unlabeled items in an iML setting.

3.1 Unknown Unknowns and Prediction Errors

Attenberg et al. [6] and Lakkaraju et al. [32] use a model's confidence score to categorize prediction errors into known unknowns and unknown unknowns. Known unknowns are errors for which the model has low confidence, and correcting such errors is useful for fine-tuning the performance of the model. Unknown unknowns are errors for which the model has high confidence, and such errors can be potentially disastrous depending on the problem domain. These errors are *unknown* to the human and the model because they happen on unlabeled data, and the model does not identify them as suspicious.

Another approach is to characterize prediction errors as feature blindness, ignorance, mislabeling, or learner errors [44]. Feature blindness errors arise because the model does not have the appropriate feature representation to learn from the training items. In contrast, ignorance errors will be correctly classified if they are added to the training set as the model already has the appropriate feature representation. Mislabeling errors come from mislabeled items by humans, whereas learner errors refer to issues caused by the configurations of the learning algorithms. In our work, we focus on finding *feature blindness* errors since these errors represent missing concepts in the current model. Feature blindness errors can be important sources of feature ideation in iML, particularly when a model is built by adding features incrementally as opposed to having access to a set of constant features (e.g., bag-of-words (BoW)).

3.2 Searching for Items to Label

There are two types of strategies for searching for items to label: machine-initiated and human-initiated. The machine-initiated (or active learning) approach uses learning algorithms to suggest

items for humans to label so that the model needs fewer training items to perform better [56]. Uncertainty sampling is one of the most commonly used active learning strategies. In binary classification, this strategy samples items whose confidence scores are near the decision boundary [35, 36]. Active learning strategies are not suitable for finding unknown unknowns or feature blindness errors because they often rely on the model's current training results, which cannot overcome the model's blind spots [6, 32, 44]. Other algorithmic sampling strategies such as diversity sampling (i.e., look for items far away from the current training set) or density sampling (i.e., focus on low density regions) [23] work with a fixed feature space; they cannot be used for finding items for which the current model does not have the appropriate feature representation, or the feature space needs to be sufficiently large which requires a large number of labels for effective feature selection.

Recently, Lakkaraju et al. introduced an algorithmic approach to find unknown unknowns directly [32]. Their approach leverages systematic biases that are concentrated on specific places in the feature space. Their explore-exploit strategy partitions a dataset based on similarities in feature space and confidence scores, and searches for partitions with high concentration of items with a high confidence. The underlying assumption for their system is that any unknown unknowns introduced to the algorithm can be characterized by the automatically extracted features (e.g., bag-of-words (BoW)), but using such features for training can have the undesired consequence of losing interpretability [25]. In our work, we aim to preserve semantic meanings during exploration to maximize interpretability.

The human-initiated approach allows humans to find the items that the model should learn. Attenberg et al. introduced the notion of *guided learning* [7] and implemented *Beat the Machine* (BTM), where crowd workers are tasked with finding items from the open world that the model is incorrectly predicting [6]. Their results showed that BTM found more unknown unknowns as compared to a stratified random sampler. Guided learning puts a significant load on humans to find adversarial items since recalling is difficult, but when combined with other active learning strategies, this method can help the model learn quickly, especially on skewed datasets. One unique characteristic of BTM is that it leverages the open world as a search space, rather than a closed and biased sample set used in traditional settings. However, this approach does not keep the concepts or structures created and leveraged by users during the search process, so it has limited support for subsequent searches.

In AnchorViz, humans do not play a passive role as labelers of items presented by algorithmic techniques but take advantage of algorithmic techniques. For example, AnchorViz uses hierarchical clustering (HC) [60] to reduce the user's search space, but ultimately, it is the user that chooses to label the item. The system also computes the similarity between sets of items in the BoW feature space and presents the results for users to take actions. In addition, we allow the users to externalize their concepts or structures so that they can decompose the target class, reuse previously defined concepts, or redefine and evolve their search strategies as they explore the dataset.

3.3 Semantic Text Representations

Having the appropriate representations (i.e., features) is critical in training a classifier because the learner cannot possibly search the hypothesis space (i.e., a space of all possible classifiers) for a solution in the absence of a learnable classification function that can correctly predict the training set [15, 44]. Thus, feature engineering—a process of producing appropriate representations that the learner can understand – is often thought of as an art and a key factor in what makes or breaks an ML project [15]. Integrating feedback from humans in the feature engineering process has been known to improve the classifier performance [49]. Earlier works have supported human-driven feature debugging and engineering [21], crowd-sourced feature labeling [12], and feature ideation [8]. When the feature representations are semantically meaningful or understandable, humans can

manipulate [63] them directly to improve the model or adjust them to debug the model efficiently [31]. Two types of semantic, word-based representations for text classification are bag-of-words (BoW) and dictionaries. While representations are typically used only as features for training the model, we extend their use case by directly using them as semantic models with which to explore (i.e., search and filter) the dataset.

BoW is a model that represents the occurrences of words in a set (or a bag) independent of how they appear in text [43] and has been shown to be successful in experiments and in practice [54]. The dimensionality (i.e., the number of words) in the resulting BoW can become too large (e.g., hundreds) for humans to retain and analyze even after common feature selection techniques [1, 69] are applied to reduce its dimensionality. In addition, the resulting BoW becomes incoherent to humans because the context surrounding the words (e.g., paragraphs, sentences) and the word orders are discarded [54]. Despite these drawbacks, BoW is commonly used for its simplicity because it can be pre-computed and can represent any text as a computable vector, where similarity or average can easily be computed for analysis [42]. In our first study, we use a fixed BoW (i.e., the list of words was precomputed and never updated through interaction) for exploring the dataset and clustering the dataset.

Dictionaries or lexicons are sets of semantically related keywords that can be generated and manipulated manually by the domain expert [63]. Even though the humans are left with the burden of creating the dictionaries or recalling all possible words that belong to a dictionary, dictionaries provide a way for humans to express a concept. For example, a dictionary containing the words "apple, orange, banana" represents a very specific *fruits* concept that is aware only of the occurrences of these three defining words. In our second study, we experiment with dictionaries as tools for both exploring the dataset and as features for training the model.

3.4 Interacting with Semantic Concepts

Categorization, or *classification* in ML terms, is an activity to determine if a specific item is a member of a category [59]. Categories can be thought of as the classes or groupings of similar items, and the concepts are mental representations of these classes that encode how these items are similar in some way. Knowing that a certain item belongs to a category can help generalize its characteristics based on your knowledge about the category. Therefore, concepts are powerful tools that allow us to infer beyond what we know about the item. One of the many views of concepts is the family-resemblance view: a concept is represented by a list of attributes or features where some may be of more importance or weight than others [51]. A schemata view, on the other hand, presents "a structured representation that divides up the properties of an item into dimensions (usually called slots) and values on those dimensions (fillers of the slots)" [45]. Understanding the attributes that define a concept (i.e., feature list), how the attributes are measured and evaluated (i.e., feature dimensions and values), and the relationships between the attributes (i.e., feature importance or correlation) are important parts of the categorization or classification task. We take inspirations from these cognitive models to design new interactions and affordances that match human cognitive processing of concepts. Specifically, our visualization allows users to define concepts through examples or keywords, visualize concepts along their similarity or value dimensions, and define a topology or layout for inspecting the relationships between concepts.

3.5 Visualization for ML

Visualization is an effective and important strategy for enabling the humans to interact with the ML algorithms and the underlying dataset, to make appropriate decisions and actions to improve the model, and to gain better understanding of the dataset and model [38, 40, 66]. There have been several visualizations proposed and used throughout the model building process, supporting

tasks such as data exploration and processing, feature selection, and model analysis, selection, and evaluation [38, 40]. Interactive visualizations can help users directly influence the output model by selecting features [8, 28], combining and tuning models [64], evaluating and diagnosing the model's performance [4, 13, 39, 50, 70], or manipulating clusters [34] or trees [65]. In addition, they can help increase the understanding of the models through, for example, providing explanations for black-box models [29] or describing the inner-workings of an ML algorithm or process [42]. However, existing work mostly focuses on understanding the current model, manipulating the model given a fixed training set, or discovering errors in the training set with little attention to discovering previously unknown errors in unlabeled items.

Visualization provides an important means for humans to interact with and understand data by mapping data to visual representations (e.g., position, color, shape) [9]. A good visualization provides an appropriate interaction language and data representations such that it becomes easy to make sense of the data and gain actionable insights. In AnchorViz, since the main user task is to locate previously unknown errors in the unlabeled items, we focus our attention on techniques for laying out and interacting with the multi-dimensional dataset.

Algorithmic approaches to laying out a multi-dimensional dataset on a two-dimensional space [37] include Principal Component Analysis [26], t-Distributed Stochastic Neighbor Embedding (t-SNE) [41]), force-directed graph algorithms [19, 42], and Sammon mapping [53]. However, these automatic approaches produce dimensions that may not be semantically meaningful or require expert understanding of the algorithms. Several works have been proposed to allow users to directly define the dimensions [27] or manipulate the parameters of the layout algorithms [24] through direct, instance-based manipulations. In AnchorViz, we allow the users to directly define the dimensions, and we support an arbitrary number of dimensions via a radial visualization [57].

Our interaction design is inspired by VIBE [47] and Dust and Magnet (D&M) [61]. D&M uses a magnet metaphor to attract similar items using pre-defined dimensions in the data. However, the existing dimensions in our iML scenario (i.e., features) may not have any connections with unlabeled items to *attract* them. Like Adaptive VIBE [2], we allow the users to create and refine *anchors* as our version of magnets. We explain *anchors* in further details in the next section.

4 ANCHORVIZ

In this section, we describe *AnchorViz*, an interactive visualization to help ML classifier developers locate prediction errors in an unlabeled dataset by using semantic concepts. AnchorViz facilitates this by letting users define concepts and find items where the model's prediction disagrees with theirs.

4.1 Design Objectives

When designing AnchorViz, we aimed at answering the following research question:

> "How might we help users use semantic concepts to discover potential prediction errors or useful concepts for a classifier through interactive data visualization?"

Discovering a prediction error in a large dataset can be like finding a needle in a haystack. This problem is aggravated by the lack of certainty of a needle actually being in the haystack at all (i.e., unknown unknowns). When looking for a rare item in a large dataset, there are many strategies one can follow, the simplest one being randomly sampling through the dataset (i.e., sifting through the haystack) haphazardly. This is an inefficient (and potentially painful) method of finding a needle. But what if we could *pull* that rare item away from the pile in which it is hidden? We use the analogy of *magnets* to shape our item-seeking solution (Figure 2). We think about different magnets to find different kinds of needles, each representing a missing concept in

the model. Since the items that are sought may be unknown, it can be difficult to automatically define corresponding magnets to pull them out.

Our work leverages this magnet analogy to generate the following design objectives that answer our research question.

4.1.1 Let Users Define Concepts of the Target Class and Unrelated Classes. To begin the exploration of the dataset for potential classifier errors, the users need to define concepts to sift and separate out the dataset. We allow users to define semantic concepts in two specific ways: example-based and dictionary-based. Example-based concepts are defined by one or more archetype examples of the concept that the user simply groups together. This is a common way for people to articulate concepts in their heads [46]. For example, a *chicken recipe* concept is defined by a set of webpages containing recipes with chicken as the main ingredient. Dictionary-based concepts are defined by a dictionary or a set of semantically related keywords that the user inputs. For example, a *firearms* concept is defined by "pistols, rifles, shotguns, and machine guns." We chose these two types of concepts as candidates for our initial design exploration and user studies for their simplicity and ease of understanding. Studies of other types of concepts, such as concepts based on trained ML models, are out of scope for this article.

4.1.2 Spread the Dataset Based on Concepts. Once the concepts are defined, the users need to see the correlation between the concept and the items to verify that their concepts are well-defined and useful by spreading the dataset based on the presence or the lack of the concepts. We call the artifacts, that define a specific concept and pull items containing some representation of the concept towards it, as *concept anchors* (or *anchors*, in short). The relationship between the anchor and each item is determined by a function that takes anchor properties (e.g., items, keywords) as inputs and outputs a score in a continuous space or an order in an ordinal space. This function defines an anchor. For instance, anchors can be defined by any model (a function that takes data and outputs predictions) that the user has previously built or any externally trained models (e.g., sentiment analysis service). Advanced users can create a custom distance function that composes multiple models and functions. Such flexibility allows trivial integration of standard active learning algorithms and opens several opportunities to support a richer interaction language and expressiveness for anchors. An anchor can be fully transparent (e.g., number of times the term "apple" appears) or fully opaque (e.g., score output of a third-party sentiment model). Regardless of how the anchor is defined, the user can interact with one or more of these anchors to gain insight about the dataset as well as the anchors themselves.

For example-based anchors, we use BoW to automatically turn items into numerical vectors and compute concept dimension as the cosine similarity in the BoW space between the concept-defining examples and other items. Intuitively, example-based anchors pull items that share more words with the concept-defining examples than items that share fewer words. For dictionary-based anchors, we compute the concept dimension as the number of times the dictionary keywords occur in the items. Intuitively, dictionary-based anchors are a form of basic keyword search where items that contain more of the keyword are separated from items that do not contain the keyword (which would remain unaffected just as the hay would remain unaffected when a magnet is brought to it).

4.1.3 Show How User-Defined Concepts Impact the Positions of Items. The users can spread the items with anchors, but to see the relationship between several concepts with respect to the items, we need to define a topology or a layout where the positioning of the items represent the relative importance or influence of each concept to the items. We aim to define a topology such that an item that is equally related to two concepts is positioned in the middle of the anchors while an item that is relatively more influenced by one concept sits closer to that anchor in relation to the other

anchor. For example, an item is *more like* catering and *less like* restaurant menu if it is attracted by one and not the other. An item is unrelated to both if it is not attracted by either anchor. We introduce the precise mathematical formulation of the layout in the later Section (4.2.3).

4.1.4 Provide Information about the Model's Current Prediction along the User's Labels. To help users find prediction errors, we need to involve the classification model being built into the exploration process. By surfacing the model predictions and the labels on the semantic topology defined by concepts, users can contrast the predictions with concepts and look for prediction disagreements in two different ways (also see Figure 10). First, the user can look for items that are predicted to be in the opposite class as the class expected to be close to a concept. For example, restaurant menus should be treated as a negative concept for the recipe class. If an item near a restaurant menu anchor is predicted as positive, then the item is a potential prediction error. Second, if an item is predicted or labeled as positive in an area of negatively predicted or labeled items, that item is an outlier worth inspecting.

4.1.5 Optimize for Efficient Reviewing Process. After a user defines a topology with anchors (concepts) and decides on a set of filters, there can still be hundreds of items to inspect to find prediction errors. We aim to support the users in efficiently sifting through and inspecting items and their predictions. Among the many ways to approach visual clutter, we explore two options: clustering and subsampling. The complexity and the number of items displayed in AnchorViz can be reduced by pre-processing the dataset and grouping similar items under a BoW embedding into hierarchical clusters. Another way to reduce the number of items is to subsample the dataset. Any selection algorithm can be applied here, and we explore a mixed-initiative approach where an active learning algorithm (e.g., uncertainty sampling) is used for item selection as it is traditionally used to reveal items where the model is likely to be uncertain about. We describe the algorithms for clustering and subsampling in detail in later Sections (4.2.5, 4.2.6).

4.2 Interface and System Design

We now describe the core design and functionalities of AnchorViz, and in the subsequent Sections (5, 6), we illustrate how the visualization was embedded in two different iML contexts.

4.2.1 Main Interaction Area. The main interface for AnchorViz (Figure 3(A)) contains a RadViz-based visualization that shows both labeled and unlabeled items as glyphs inside the ring (Figure 3(B)). The main area also shows anchors representing concepts (Figure 3(C)) on the ring. Without any anchors, the visualization displays all items, or glyphs (Figure 3(D)), at the center of the ring. As users define concepts, the item's layout will change to reveal affinities between items and concepts. This main visualization area is accompanied by a legend area (Figure 3(E)) to help users differentiate and focus on different types of glyphs.

4.2.2 Anchor Definition through Examples and Dictionaries. As we discussed earlier, we designed and implemented two types of the anchors that the users can define (4.1.1): example-based and dictionary-based. To create an example-based anchor, users can drag an item to the ring. Users can also drag an item to an existing anchor to update the corresponding concept. A user can click on an anchor to reveal the anchor details in a dedicated area (or a dialog can be used as an alternative) of the interface that hosts AnchorViz (Figure 3(G)). There, users can provide a meaningful name for the anchor, view the list of items that belong to the anchor, and remove items that no longer belong to the anchor.

To create a dictionary-based anchor, users can click on the ring to bring up a dialog that solicits the dictionary name and keywords that belong to the dictionary (Figure 3(H)). Like example-based

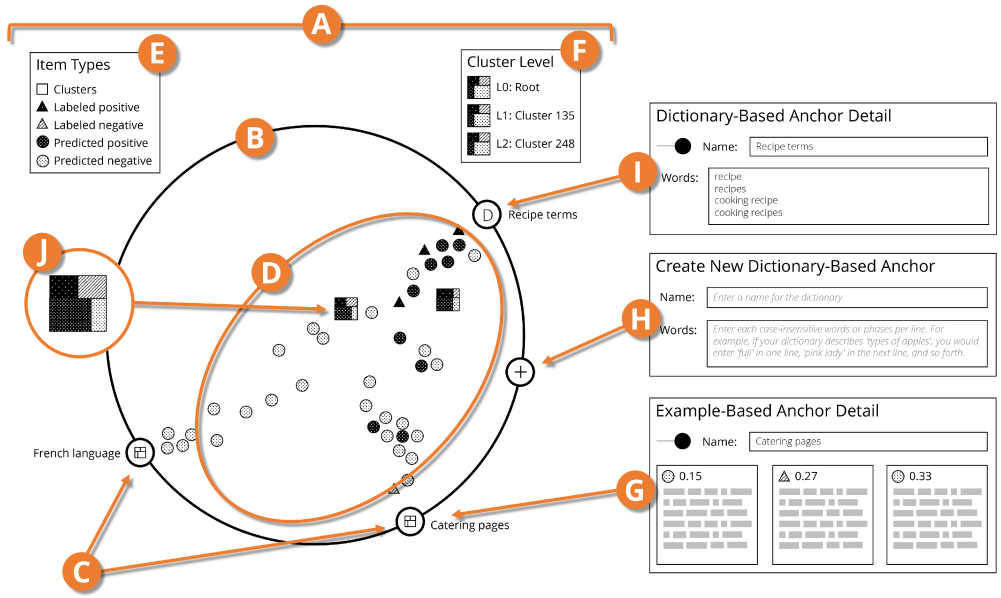

Fig. 3. AnchorViz wireframe. The main interface (A) contains the visualization and shows data points (D) within the ring (B) on which the anchors (C) are positioned. The legend for data points (E) also acts as filters. The navigator (F) shows which cluster the visualization is displaying in the current navigation stack. Clicking on an anchor shows a dedicated interface for its details appropriate for its type (G, I). Clicking on the ring shows a dedicated interface for creating a dictionary-based anchor (H). Clusters are represented as treemap-style squares (J).

anchors, users can click on an anchor to reveal the anchor details where they can modify the dictionary or hide the anchor from the visualization (Figure 3(I)).

4.2.3 Layout and Anchor's Manipulation. Once users create an anchor, they can see the correlation between the concept represented by the anchor and the items (4.1.2) or the relationship between several concepts with respect to the items (4.1.3). When users move the anchor along the ring, they see items follow concepts that they are related to. We encode the relative similarity between items and the surrounding concepts into their position inside the ring. We do this using a non-orthogonal coordinate system, defined by the center point of the ring and the axis passing through each anchor. Namely, an axis k is a vector with the length of the outer circle's radius r and an angle θ to the corresponding anchor (Equation (1)):

$$\vec{V}_k = (r \cdot cos\theta, r \cdot sin\theta). \tag{1}$$

An item along an axis forms a vector with an angle identical to that of the axis and a magnitude of the cosine similarity in the BoW space between the item and the items or words defining the anchor. Given a set of anchors, then the position of an item inside the ring is the sum of the vectors to each anchor (Equation (2)):

$$Position(i) = (x_i, y_i) = \sum_k \frac{value_k(i)}{\sum_j value_j(i)} \cdot \vec{V}_k \tag{2}$$

This formulation satisfies our design goals (i.e., the items that are closer to an anchor are more like the items or words defining the anchor). Items that are attracted to an anchor will move along with it, whereas items dissimilar to an anchor's concept will remain still (4.1.3). By creating and

manipulating anchors, users are effectively creating a topology in the semantic space defined by the anchors' concepts. Since we want to ensure all items stay inside the ring, we normalize an item's value on an axis by the sum of all its values on all the axes. This normalization follows the typical normalization used in RadViz [22].

While there are other techniques for visualizing the aforementioned topology, we chose RadViz because of its support for an arbitrary number of axes and its flexibility in the positioning of axes while preserving the relative independence of the axes [57]. In this article, we do not explore these alternatives and leave them as opportunities for future work.

4.2.4 Contrasting Model Predictions against Concepts. We encode model predictions and user labels into the color and shape of items as illustrated in Figure 3(E) such that the users can leverage their visual perception to quickly find prediction errors or inconsistencies between predictions and their expectations based on the concepts at hand (4.1.4). As displaying many items can become visually complex, we provide users with the ability to filter the types of items displayed inside the ring. Users can do this by toggling categories in the legend area near the ring.

4.2.5 Hierarchical Clustering. We cluster items in the dataset based on the cosine similarity in the BoW space as the distance between any two items. The distance between clusters (C_a and C_b), including single-point clusters, is the average distance between pairs among two clusters (Equation (3)):

$$distance(C_a, C_b) = \frac{\sum_i \sum_j distance(C_{ai}, C_{bj})}{|C_a||C_b|} \tag{3}$$

We group items using a hierarchical clustering algorithm. In each step, the algorithm selects a pair of items with the shortest distance and merges the two as a new cluster. The step is repeated until there is only one cluster left. We reorganize the result of the hierarchical cluster, which is a binary tree, into a n-ary tree from the top tree node and expand the tree in a breadth-first order. At any given node of the n-ary tree, it can contain leaves (items) or sub-trees (clusters) which can further divide into sub-trees and leaves. For our user study, we used $n = 500$ which is neither too cluttered nor sparse. We leave choosing an optimal value of n for future work.

The groupings built using this algorithm help reduce the search space items by limiting the number of visible data elements and allowing users to review groups of similar items rather than individual items. These groups or clusters do not change in composition when anchors vary in definition or location, since they group items close in a BoW semantic space, not items that are visually close according to the anchor topology. However, clusters move along with the anchors that they are most similar to, just like any individual items. A cluster's position is based on the average similarity of all the leaf items inside the cluster.

We display these pre-processed groups of similar items (hereafter, we refer to a group of similar items as a *cluster*) as treemap-style squares (Figure 3(J)). The size of the square is a function of the number of items inside the cluster. Each square is a single-level treemap [58] representing the composition of items in four categories (labeled positive, labeled negative, predicted positive and predicted negative). By using treemaps as glyphs, we aim to provide an at-a-glance understanding of the distribution of items within so that users can make a quick decision to navigate into the cluster. For example, if the cluster contains predicted negative items and labeled positive items, there may be potential inconsistencies between the clustering, the classifier, or the labels that need to be investigated and resolved. We display the same treemap for each anchor as a way to visualize their item composition as well as their class association along with the descriptive name.

4.2.6 Subsampling. We sample at most 100 items where the uncertainty sampling picks the top 50 predicted positive and 50 predicted negative closest to the decision boundary. Unlike

hierarchical clustering which is a pre-processing step, subsampling can be done in real-time with every iteration of classifier training. With this type of visual reduction, we can render a simpler visualization without clusters or a navigation affordance.

4.2.7 Interacting with Items and Groups. When users click an item, the corresponding interface reveals the details of the item in a dedicated area. This item detail area lets users inspect the item, its label, and the model's prediction. It gives users a quick way to check if the model predicts the item correctly and provide a correct label (4.1.5). If example-based anchors are available, the interface also provide users with the ability to create an anchor from the details of an item.

Users can navigate into a cluster, if present, by clicking on it, and the visualization will update to show the items and clusters that are children of the cluster they clicked. We provide users with a navigation affordance (Figure 3(F)) that shows the current navigational path inside the hierarchical cluster structure. The root cluster at the top of the path or stack represents the entire dataset. Users can navigate back to a previous level through the navigator or double click on any empty area inside the ring.

4.3 Implementation

Our system consists of two components—a self-hosted web service and a web application. We implemented the web service using .NET, ASP.NET Core and the Nancy framework; we implemented the web application using Typescript, React, and D3. All relevant data, including the exploration dataset, training data, client states, and classifier configurations, is persisted to disk on the web server to support browser refreshes and for evaluation of the study results.

5 USER STUDY 1: CLASSIFIER ERROR DISCOVERY

We conducted a user study to evaluate our visualization and its effectiveness in helping users to discover classifier errors. To the best of our knowledge, AnchorViz is the only visualization tool for error discovery that allows interactive exploration with the users' explicit semantic formulations, so there is no baseline to compare purely on its contribution to facilitating data exploration. Thus, the focus of our study is to understand the different strategies people use to explore the dataset and observe people's interactions with the anchors while all other conditions (e.g., features, distribution of positives) are fixed. This section describes the design of the user study and data collection, and we introduce definitions of several metrics that can be used for analysis and comparison in the future.

5.1 User Study

5.1.1 Study Design and User Task. To evaluate the visualization, we provided the participants with a specific scenario. The premise for the user study is that the participant is developing a binary classifier using an iML tool (i.e., iteratively sampling for items, labeling the items, adding features, and debugging the target classifier). Through this iterative process, the participant has achieved almost 100% accuracy on the training set. Despite high training accuracy, the classifier performs poorly on a held-out test set, and the participant does not know *a priori* any items or features where the classifier could be making a mistake. Therefore, the participant switches to explore a large unlabeled dataset to find potential sources of errors.

Based on the above scenario, we asked participants to (1) explore the unlabeled set to find items where the classifier is making a mistake, (2) find a set of items that are diverse from each other, and (3) try to understand the dataset and classifier performance in the process.

Even though the study is set up as an iML scenario, we disabled featuring and interactive training of the model. The goal of this study was not to observe the compounding effect of choosing the

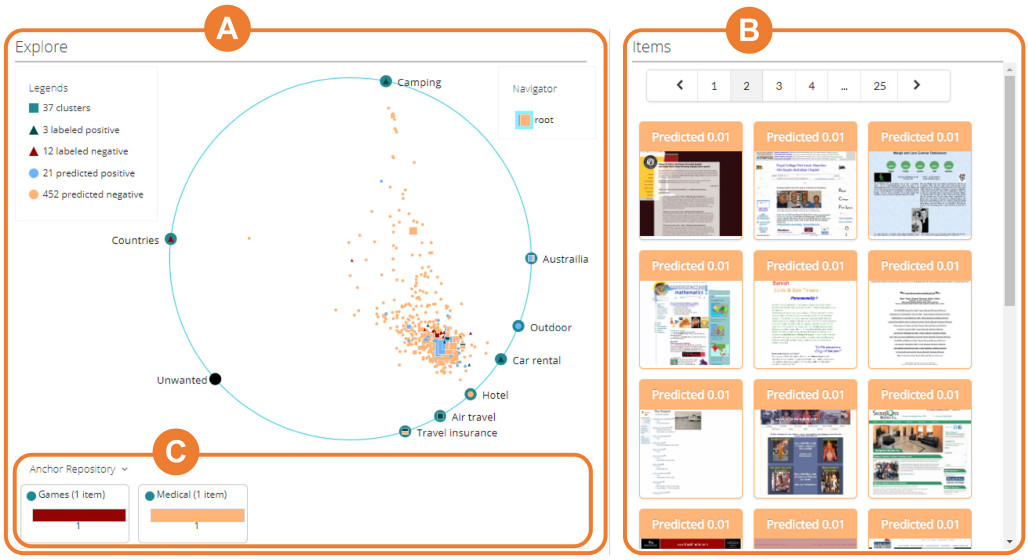

Fig. 4. Overview of the first user study interface. The interface contains the Explore pane on the left which hosts AnchorViz (A) and the Items pane on the right (B) which shows a paginated grid of thumbnails of all currently visible items in the visualization. Anchor repository (C) contains unused anchors.

right item and retraining the model to inform the next sampling activity. Instead, the goal was to evaluate the quality of the selected items. Therefore, we only allowed interactive sampling and labeling.

5.1.2 Interface Design. For the first study, we integrate AnchorViz in an interface that uses example-based anchors and hierarchical clusters and supports an exploration experience designed specifically for webpage data. The interface (Figure 4) is divided into two main areas: Explore pane on the left and Items pane on the right.

The Explore pane (Figure 4(A)) contains the main AnchorViz visualization, and is the place where users explore and select items from the dataset. Below the visualization, there is an Anchor Repository (Figure 4(C)) containing anchors that the user removes and are available for reuse. The Items pane (Figure 4(B)) shows a paginated, grid view of the items currently displayed in the main visualization. When users click an item, the Items pane (Figure 4(B)) will switch to show the detail of the selected item, cluster or anchor.

5.1.3 Dataset, Classifier, and Setup. To set up the user study according to the scenario above, we built a binary classifier whose training accuracy is 100% in an iML tool. The binary classifier outputs prediction scores from 0 to 1 and uses 0.5 to be the decision threshold. For our dataset, we took webpages and categories from the Open Directory Project (http://www.dmoz.org/) where the webpages were voluntarily organized into hierarchical categories by the community editors. After rendering the webpages, we discarded the webpage markup and retained only rendered and visible text tokens for each webpage. Following the categorization descriptions provided by the dataset, we built binary classifiers for several hours in an iterative model-building loop using logistic regression as the learning algorithm, text search and uncertainty sampling for item selection, and human-generated dictionaries (single weight for a group of n-grams) as features. Out of nine binary classifiers, we picked a classifier for predicting *cooking*-related webpages which had the highest train accuracy (97.5%) with 674 labeled items and 37 human-generated features. The

choice of the learning algorithm is independent of the problem of discovering errors and is out of scope for this article.

To control for the ratio between the labeled set and the unlabeled set, we uniformly sampled for 4,000 items from the full dataset (50% positive for cooking) and 400 items in the labeled items such that approximately 8–10% of the dataset was labeled. To simulate blindness to concepts within positive and negative classes of cooking, we removed 25 features related to cooking (e.g., appliances, seasoning), and we left only one n-gram in each feature to degrade its test accuracy further. After training the classifier, we subsequently removed incorrect items to achieve 100% training accuracy. The final cooking classifier had 309 labeled items (130 positive), 12 features, 100% train accuracy and 75.8% test accuracy. This starting point emulates an early stage of an interactive session where a user builds a classifier by providing both labels and features. For participants to practice, we also picked a *travel*-related binary classifier to use throughout the tutorial. This classifier only appeared during the practice round.

A limitation of the current interface design is that the visualization requires at least one anchor to be present to begin the exploration process. In an ideal case, we would provide the users with ways to bootstrap the visualization such as selecting items based on keyword search or choosing a set of labeled items to seed an example-based anchor. Evaluating the cold-start scenario is out of scope for this study, and therefore, we bootstrapped the visualization with one pre-defined anchor containing a positively labeled item and another containing a negatively labeled item.

5.1.4 Participants. We recruited 20 participants from a large software company. We used four participants for the pilot study, and we used data from the remaining 16 participants (7 female) for the analysis. We categorized participants into four groups according to their ML background. Four participants took some courses in ML (P1-4), four occasionally build ML models in practice (P5-8), three frequently build ML models (P9-11), and five have a doctoral degree in ML-related fields (P12-16). Our participants' professional roles in the company are that of applied data scientist (9), program manager (2), researcher (1), financial manager (1), and software engineer (3).

5.1.5 Procedure. We conducted our study through video conferencing sessions which we recorded. During the sessions, participants shared their screens and thought aloud while performing their tasks. The user study consisted of four parts: The first part took about 20 minutes and involved an introduction to basic ML knowledge such as classification, errors, precision and recall, description of the data set, overview of the study interface, and introduction of the travel class (webpages about traveling and tourism). The second part took between 10–20 minutes and was a practice round using the travel class as a reference to get familiar with the interface, followed by an introduction of the cooking class (webpages about home cooking) which was to be used for the actual task. The third part was the main study task and took about 20 minutes. During this part, we asked the participants to use AnchorViz to find and label a diverse set of items where the participants disagreed with the classifier's predictions. At the end of the study, we asked participants to complete a small survey to assess their satisfaction and to collect open-ended feedback. Each study session lasted between 70–90 minutes. We compensated each participant with a $30 gift card.

5.2 Quantitative Data Analysis

We focused our analysis on the items that the participants discovered and the circumstances and the behaviors surrounding their discovery. We also instrumented the web application for user behavior to replay their interactions with anchors or items. Since we are not studying the effectiveness of user interactions on the model performance, we do not report the usage statistics.

5.2.1 Anchor Effectiveness. To measure the effectiveness of anchors on discovering errors, we define two metrics: *anchor error precision* (AEP) and *anchor error recall* (AER). The two metrics have similar concepts as the common precision and recall metrics in ML, but instead of measuring based on the target classes (positive/negative), AEP and AER look at whether items are errors or not.

Before further defining AEP and AER, we first define *contrasted items*. Contrasted items aim to capture items that become salient through anchors. The use of contrasted items is a form of automatic discovery once the anchors are given. The steps to determine whether an item is a contrasted item are:

(1) Collect neighbor anchors: Given an item, take up to three of its nearest neighbor anchors. Note that an anchor must be within $r/2$ range of an item to be its neighbor.
(2) Determine the class label for each neighbor anchor: For every neighbor anchor, we determine its label based on majority voting from its items' ground truth labels. That is, if an anchor has three items, with two of their ground truth labels are positive and the other is negative, then the label of this anchor is positive.
(3) Determine the contrasted base label: The contrasted base label is determined by majority voting from all the neighbor anchors' labels. The contrasted base label is unsure in a tie.
(4) Determine if the item is a contrasted item: If the item's predicted label is opposite to the contrasted base label (not including unsure), then the item is a contrasted item.

Then we can define AEP and AER as follows:

$$AEP = \frac{\text{\# true errors in contrasted items}}{\text{\# contrasted items}} \tag{4}$$

$$AER = \frac{\text{\# true errors in contrasted items}}{\text{total \# true errors}} \tag{5}$$

An intuitive explanation of these two metrics is that they help examine how many error items a setup of anchors can bring into attention. Note that we calculated these two metrics based on all of the contrasted items given the whole layout of anchors and points at a time, not merely a single anchor. We also consider all items by their spread-out positions, not their clusters' positions, as our goal of the two metrics is to measure the anchors' effectiveness, not the clustering algorithm.

We calculated AEP and AER for all active anchor settings of each participant over time. By active anchor settings, we refer to the layout of anchors and items when participants actively interact with items, which include: creating an anchor, adding/removing items in an anchor, navigating into a cluster, viewing an item, and labeling an item. Thus, if a participant interacts with 10 items, the participant will have 10 measurements for AEP and AER.

Figure 5(B) shows the average AEP and AER of each participant. The reference line "Random" of AEP shows the baseline for AEP to be 0.26, the error rate in the dataset. This baseline is how likely an item is an error if randomly selected by a user. As the figure shows, 12 out of 16 participants had an average AEP greater than random. This indicates that contrasted items are likely to pull errors to attention.

For AER, we plotted the average across participants as a reference line since the same analogy in random is 1 (all the errors are in the dataset). The average AER (0.13) indicates that only about 13% of the errors are the contrasted items. However, the goal of our tool is not to find all errors, but to find errors that are critical. We further analyzed the types of errors in contrasted items and found most errors in contrasted items have a higher chance to find feature blindness errors than other approaches, including the algorithmic samplers, as we described in Section 5.2.3.

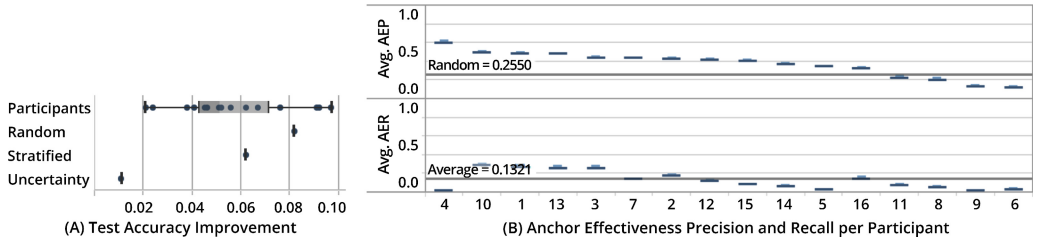

Fig. 5. (A) Test accuracy improvements across participants and algorithmic samplers. All participants built better performing classifiers than an uncertainty sampler, and three participants built better performing classifiers than random samplers. While random samplers performed well on the test set, they discover less feature blindness errors as indicated by Figure 6(B). (B) Average anchor effectiveness metrics (AEP/AER) for each participant. Most participants have better AEP than random.

5.2.2 Error Categorization and Generalization. We evaluated the effectiveness of our visualization on discovering prediction errors in two ways. First, we computed the number of prediction errors that the participants discovered. Then we examined whether these errors were feature blindness errors by individually retraining the classifier with every item and seeing if that item was still a prediction error. Second, we looked at the score distribution of the discovered items to see how many high confidence errors the participants were able to find. We computed the magnitude of error as the absolute difference between the prediction score and 0.5. For comparison, we computed the same metrics with other samplers (e.g., uncertainty, stratified random samplers) given a fixed number of items comparable to that of the participants.

We also measured the quality of items by evaluating a classifier trained with the discovered items against a held-out test set. There are several challenges here. One is in simulating a human's featuring ability which is required in an iML loop that we operate in. Another is that the test set may not include a representative item for a concept that the participant discovered. For example, when building a cooking-related webpage classifier, it can be the case that there is not a single item with a reference to "insect recipes" (a positive concept) in the test set which can be found during exploration of the unlabeled set. Fully acknowledging the challenges in evaluating the classifier's generalization performance, we computed the classifier's accuracy on the held-out test set using the original feature set (37 features). Our justification is that comparing two classifiers with or without the discovered items on a fixed feature set would provide us with a glimpse into the quality of the items added.

5.2.3 Error Analysis. For the analysis of the items discovered by the participants, we used the ground truth labels provided by the original dataset as well as 4000 exploration candidate items (50% positive) and 1600 test items (50% positive).

On average, participants discovered 40.9 errors ($\sigma = 19.6$). We looked at the percentage, instead of the count, of errors in the total number of discovered items because of high variability in the number of discovered items. We also looked at the first 50 (mean number of items found by participants with doctoral degrees in ML) items returned from algorithmic samplers. Of the unlabeled items (3691 items), the initial cooking classifier made prediction errors on 943 or 25.5% of the items and feature blindness errors on 886 or 24.0% of the items. Except for P10 who discovered errors at a rate of 24%, all participants discovered errors at a higher rate than the total error distribution, random, stratified random, or uncertainty sampler. The uncertainty sampler selected errors at 47.8% but only 8.7% of the sampled items were feature blindness errors. Again, except for P10, all participants discovered feature blindness errors at a rate higher than any algorithmic samplers. Figure 6(A) shows the distribution of errors among discovered items across participants.

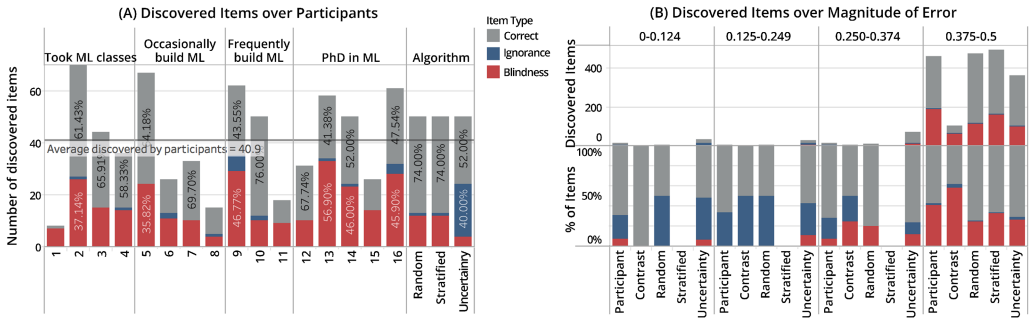

Fig. 6. (A) Distribution of items across participants and algorithmic samplers. Text in each bar shows the percentage of items in different item categories. All participants except for P10 discovered feature blindness errors at a rate higher than any algorithmic samplers. (B) The total number of discovered items (top) and the percentage of items (bottom) in different item categories and in four error magnitude bins. The distribution is skewed towards high magnitude errors (0.375–0.5). Among high magnitude errors, participants discovered and contrasted more feature blindness errors than algorithmic samplers.

Figure 6(B) shows the distribution of the magnitude of errors across different samplers. Here, we looked at all the unique items discovered by the participants ($n = 493$) and contrasted items (defined in Section 5.2.1) among the discovered items ($n = 111$), and we selected the first 493 items from each of the algorithmic samplers. According to the distribution of discovered items, the dataset is skewed towards high confidence scores and high magnitude of error. However, the percentage of errors indicates that the items selected by participants and the contrasted items had a higher chance of discovering errors than algorithmic samplers. The uncertainty sampler selects more ignorance errors than any other samplers. This was expected because uncertainty sampling selects examples that are ambiguous for the existing features. Discovering a feature blindness error can be considered as *blind luck*.

5.2.4 Test Accuracy. While keeping the features fixed to the original set of 37 features, we compared the classifier accuracy on the held-out test set before and after the discovered items were added to the training set. As before, we selected first 50 items from algorithmic samplers for comparison. Figure 5(A) shows that all participants made test accuracy improvements, and their classifiers performed better than the uncertainty sampler. Because the dataset is skewed towards high confidence scores, random samplers were able to select high magnitude error items and improved test accuracy. However, random samplers discovered fewer feature blindness errors as indicated by Figure 6(B), and three of our participants (P9, P13, P16) had classifiers with higher test accuracy than the ones created by random samplers.

5.3 Qualitative Analysis of Participant Exploration Strategies

We captured the participants' interaction and behavior through recordings of their conversation and usage of the tools. We coded their actions (e.g., click cluster, move anchor), the motivations for their actions (e.g., inspect prediction inconsistencies, see how items move), and the reactions to the actions they performed (e.g., whole cluster is positive, anchor is not very good). We used the results of the qualitative analysis to catalog different exploration strategies used by the participants and insights they obtained.

We outline different exploration strategies that the participants used when interacting with the visualization tool from qualitative analysis. We focus on three key aspects of the participants' use: (1) their reasons for creating anchors, (2) their exploration strategies, and (3) their understanding of classifier performance.

5.3.1 Reasons for Creating Anchors. All participants created anchors when they discovered a concept that could be useful for defining positive or negative classes (e.g., desserts, biography, Turkish cuisine). Anchors were also used to capture items that could potentially be confusing to the classifier. For example, P16 found a webpage containing a recipe, but a majority of the webpage was dedicated to explaining the benefits of a certain ingredient. P3 found a personal blog webpage of someone who enjoys eating food, but the webpage was not about cooking. One participant (P8) created anchors to group the items based on the confusion categories (i.e., false positive, false negative, true positive, true negative). P14 created an anchor from an entire cluster of items from a single recipe site to create a good recipe anchor. One participant (P1) tried to create anchors to differentiate webpages with recipes that did not have typical recipe headers (e.g., ingredients, instructions, prep time) from webpages with prototypical recipe structures. Some participants (P11, P13, P14) created anchors to capture or filter potential issues with the dataset. For example, there were webpages with recipes translated into two languages, webpages written only in one foreign language and are still cooking related, and webpages written in foreign languages and are not cooking related. Some participants created anchors to validate their hypothesis formed from exploration. For example, P9 discovered that the classifier was making a mistake on a webpage with lists of recipes and suspected that a lot of webpages with lists of recipes would be errors. He created an anchor to look at similar items around the anchor and said, "*basically pages that index a lot of recipes is not [correct].*"

5.3.2 Exploration Strategies. Participants used various aspects of the visualization to explore the dataset, and different strategies were used to meet their needs and at different points in time. Below, we enumerate an exhaustive list of how the participants explored the dataset.

Participants leveraged the visual encoding (i.e., position, color) to look for discrepancies or items that stood out. All participants looked at outliers in color (e.g., labeled positive in the sea of predicted negatives) or outliers in position (e.g., an item positioned away from the cloud of items in the middle. One participant (P1) looked at discrepancies between the global distribution and the local cluster distribution. Some participants used the cluster treemap to search for a specific cluster distribution (e.g., cluster with mixed predictions).

All participants, at some point, used the placement of anchors to define and modify the exploration topology. Most participants placed positively associated anchors on one side and negatively associated anchors on the other side of the visualization. P13 spread out the anchors to see items better. Some participants overlaid anchors on top of each other to combine concepts or to strengthen their effects (P7, P11, P12, P16). One participant (P3) removed anchors because he was not making any forward progress, and another participant (P16) removed anchors because he did not understand their meanings. One participant with a good understanding of BoW similarity (P9) inspected words in items both close to the anchors and far away from the anchors.

Participants used the anchors to pull or push items of specific prediction class, label class, or concept from the cloud of items. As mentioned earlier, some participants used the anchors to validate that there were a lot of non-English webpages in the dataset. Some participants used the labeled items as a way to validate that the anchors were pulling correct items. Some participants moved the anchors to create a parallax effect to see which items were impacted by the anchor or to determine the effectiveness of anchors they created. Most of the time, the participants were using the anchors to look for items near the anchors that were predicted to be in the opposite class from the labels of the items in the anchors.

As participants interacted with anchors, they refined the anchors to make them more effective. P11 added similar items to an existing anchor to make the anchor stronger. P9 and P11 removed items from an anchor because the definition of the anchor had deviated from its initial intent P4

renamed the anchor from "cucumber" to "vegetables" because the anchor's representative concept evolved over time. Most participants added items of a similar concept to existing anchors. When the participants discovered an item and they could not decide on its label, they added the item to an anchor as a way to collect similar items and to defer the decision until a later point (P12 - "*Cook book can be negative in my opinion, but I'm not sure. It's about cooking and maybe it's related to cooking. Let's treat it as positive for now.*").

All the participants navigated into a cluster at some point, but the intentions varied across participants and contexts. Some participants (P7, P13) went into clusters of a specific prediction or label class to look for good items for creating anchors. Some participants (P1) looked for clusters with discrepancies between different sources of truths (clusters, the classifier, and labels). Others (P5, P9) used the clusters to reduce the search space or to evaluate items in bulk. Once anchors were created, some others (P3, P15) navigated up and down the cluster hierarchy to see the anchor's effects at various levels.

Some participants developed a repeatable process for exploration. The process for P11 was to find negatives around positive anchors, look at the clusters of mixed predictions near positive anchors, create appropriate anchors and to repeat the steps. When the participants reached a saturation point for exploration in the current view, P9 went back to the root cluster to start over, and P3 removed anchors to reset the topology.

5.3.3 Understanding Classifier Performance. Throughout the process of exploration, some participants were able to comment on the classifier's performance. In general, participants could discover an error or a concept, explore similar items using anchors, find a whole group of errors (P9 - "*Most of the same webpage [in this cluster]. This whole cluster seems to be false negative.*"), or make general statements about the concept (P2 - "*I guess we're doing nice for sushi*"). P9 said, "*I looked first at predicted positives. They all looked to be reasonable. There's no pattern there. I concluded there's mostly likely not bad precision problem. Most likely recall problem. Looking at the negatives, indeed, it seems to be a lot of missed use cases. I found two main categories of false negatives, mostly the lists of recipes and the more instructional pages that has a lot of words in them.*"

Although our visualization did not provide any debugging support, participants were able to come up with some explanations for the classifier's mistakes. For example, P14 commented that "*cooking blogs are easy to mis-predict*" because they share a lot of words with non-cooking blogs. P10 found an item with "*poems and other mixed things*" about cooking and called it a "*grayish area.*" P4 found that webpages containing words with multiple meanings, such as "squash" (a food and a sport), could be confused with cooking. Some understandings about how the classifier is learning from tokens was also used. For example, P11 said, "*I think there are few words. That's why it's not able to classify it as recipe.*"

5.4 Exploration Areas for the Subsequent Study

When we asked the participants what they liked about the interface, participants commented that the interface supported exploration in an intuitive and fun way. P4 said, "*visual inspection of outliers is intuitive,*" and P6 said that the interface is "*very intuitive when the anchors are meaningful.*" P13 liked the "*visualization and self-organization of item space.*" P14 commented during the study, "*I play too happily with it!*" The results from our first study were encouraging, but we were left with many open research questions which we sought out to explore in a subsequent study.

5.4.1 Searching for the Appropriate Concept Representation. In this first study, we defined a semantic concept using items and their similarity in the BoW space. The use of the BoW for defining an anchor makes the interaction simple since the users rely on BoW to extract patterns; grouping a few items automatically extracts the underlying concept represented by those items. However,

this approach may not be expressive enough to correctly capture a user's intent or concept. Participants with more experience in ML quickly understood the implications of using cosine similarity in BoW (i.e., that BoW uses tf-idf) as an anchor distance metric. Other participants struggled to rationalize the proximity of items to certain anchors. For example, P11 thought adding similar items to an existing anchor made the anchor *stronger* when, in fact, the anchor was potentially covering a broader concept through additional terms being captured by BoW. P16 wanted to *merge* concepts by putting two anchors on top of each other, but instead, the action resulted in increasing the weighting of the anchors. Since example-based anchors are limited to items that the user discovers, P10 wanted "*keyword search through the data to find tougher items from intuition and see where they lie on the interface.*" Understanding the users' intent to merge or strengthen anchors is critical in designing the next iteration of the visualization and interactions. In particular, careful balancing between the simplicity of the interactions and the accuracy of the representation of the current users' mental model is a crucial design factor in this type of systems operating with semantics.

We used the above feedback and observations to inform a follow-up iteration of AnchorViz, where we explored the use of keyword or dictionary-based anchors. Among the many ways to define dictionary-based anchors, we picked the one simply defined by the number of times a bag of semantically related words appears in an item. For example, if the user defines a fruits anchor with the words "apple, orange, banana," an item that contains five occurrences of the fruits words will be closer to the anchor than an item that contains one occurrence. Dictionary-based anchors leverage human knowledge and can be useful in proactive search instead of passive discovery ("*keyword search through the data to find tougher items from intuition and see where they lie on the interface.* - P10)." This type of anchor is more transparent and intelligible to the users because the keywords in the dictionaries are explicitly defined by the users, but it loses the high-dimensionality that BoW provides especially when precise concept representation is not a priority. For instance, when participants discovered foreign language items, rather than precisely defining the French language, they wanted to roughly pull away items that contain French terms ("*it was also really useful for discovering items that were out of scope (e.g., foreign language)*" - P13). In our second study, we explored dictionary-based anchors and evaluated the tradeoffs between these two approaches.

5.4.2 Closing the iML Loop. So far, we have observed that AnchorViz helps to facilitate discovery of classifier errors and features. P13 said the interface is "*great for concept discovery and dynamic labeling.*" This interface can also be used for managing discovered concepts and "*creat[ing] new concepts that can be shared across people (P12),*" and for "*hierarchically expressible classification tasks (P16).*" In contrast to previous approaches to find unknown unknowns, our visualization also facilitates feature blindness discovery: P2 commented that this interface is useful for "*finding features I didn't know about and finding instance that can represent some useful concepts.*"

An item is deemed as good for labeling and adding to the training set for many reasons: it is a good representation for or against the target class, or it contains new information that the classifier should learn about. Once new information is discovered, labeling alone is inefficient if the classifier does not have the right representation to process the information; featuring is necessary to change the representation of the learning model. Once the feature is added or modified, the classifier needs to be retrained to evaluate whether the feature is beneficial to the generalization of the target concept. Because of the above, we argue that a complete iML tool should support all steps of selection and labeling of items, featuring, and training the model in an iterative loop.

In our second study, we integrate AnchorViz in an end-to-end iML tool to study the impact of semantic visual exploration in the model building process. Specifically, we focus on the beginning stages (i.e., cold start) of the model building process. Finding good items and features at the early

stages of model development should not only improve the efficiency but also help the users obtain a better understanding of the dataset and make appropriate next steps.

6 USER STUDY 2: FEATURE DISCOVERY AT COLD START

The two main areas for further exploration from the first user study were (a) alternative concept representation and interactions and (b) closing the iML loop. We specifically target the cold start stage of the model building experience to evaluate how AnchorViz can help users simultaneously learn about a new dataset and build their target classifiers iteratively. We focus on dictionary-based anchors to minimize the user burden we observed in interpreting the high-dimensional BoW space. Supporting the full iterative iML loop, with humans explicitly providing the labels as well as hand-crafting their features, necessarily introduces a lot of moderating variables such as individual feature ability, familiarity of the target class, intrinsic and extrinsic motivators, and the like. Nonetheless, we strive to minimize these individual differences by running a controlled, counter-balanced, within-subject user study. Our goal for the second study is to contrast two considerably different approaches to exploration: human-driven (AnchorViz) and machine-driven (uncertainty sampling).

6.1 User Study

6.1.1 Study Design and User Task. We focused the second user study on the very beginning phase of building a binary classifier. The user scenario for the study is that the participant had just gotten access to a large unlabeled dataset and is trying to build a binary classifier for a specific concept from scratch using an iML tool. We asked the participants to (1) find items and concepts to teach through exploration, (2) add labels and features to train the classifier, and (3) build the best binary classifier possible given the tool while learning about the dataset.

To observe the effect of using the visualization at this initial phase of building a binary classifier, we conducted a controlled experimental study comparing the use of the visualization (*Visualization* or human-initiated condition) with a conventional approach (*Uncertainty* or machine-initiated condition). We used the within-subject approach to control for individual differences in featuring abilities. We chose two different concepts for the classifiers and counter-balanced the concept as well as the order of the treatments to minimize the ordering and learning effect. Therefore, each participant was given two conditions with different concept for each condition.

6.2 Interface Design

For the second study, we integrated AnchorViz in an iML interface that uses dictionary-based anchors for exploration and uncertainty subsampling to reduce its visual complexity. Figures 7 and 8 show the interfaces for the two conditions that support the full end-to-end interactive classifier building experience designed specifically for text-based data.

6.2.1 Supporting the Full IML Workflow. We support all three key activities (Figure 1) within the iML workflow, and we provided consistent UIs to normalize the available functionalities in the interface. Exploration and sampling is supported in two ways: the users can select unlabeled items in the Exploration panes (Figures 7(A) and 8(A)), or the users can select labeled items that are incorrectly predicted by the model in the Errors pane (e.g., Figure 7(G)), which shows a list of training errors as small previews. Selecting an item populates the labeling area (e.g., Figure 7(C)); the user sees an item in its full textual content and can label it using the "Yes" or "No" buttons (Figure 7(D)). The button's border indicates the current classifier's prediction, and the button's fill color indicates if the user has labeled it. Words in the item that match keywords in the currently active features will be highlighted. A scrollbar beside the item shows markers for these words so

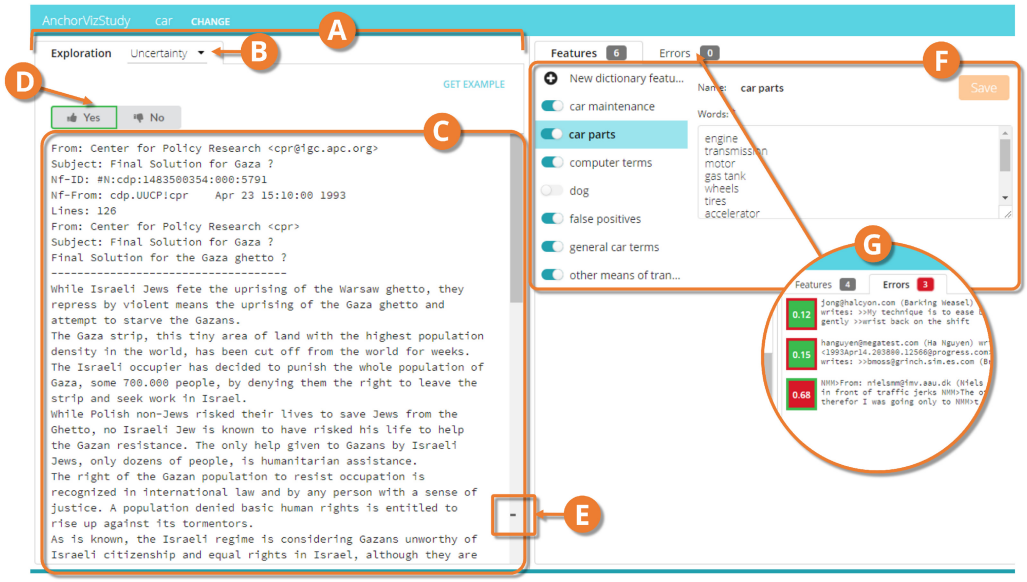

Fig. 7. The Uncertainty interface contains the Exploration pane on the left (A) which provides a drop-down (B) for choosing the sampling method and labeling options (D). The content of each item is displayed (C) with marks on the scrollbar (E) that indicate where the features occur. The right side of the interface contains the Features pane (F), where one can toggle on or off a feature or create or modify features, and the Errors pane (G), where one can inspect training errors.

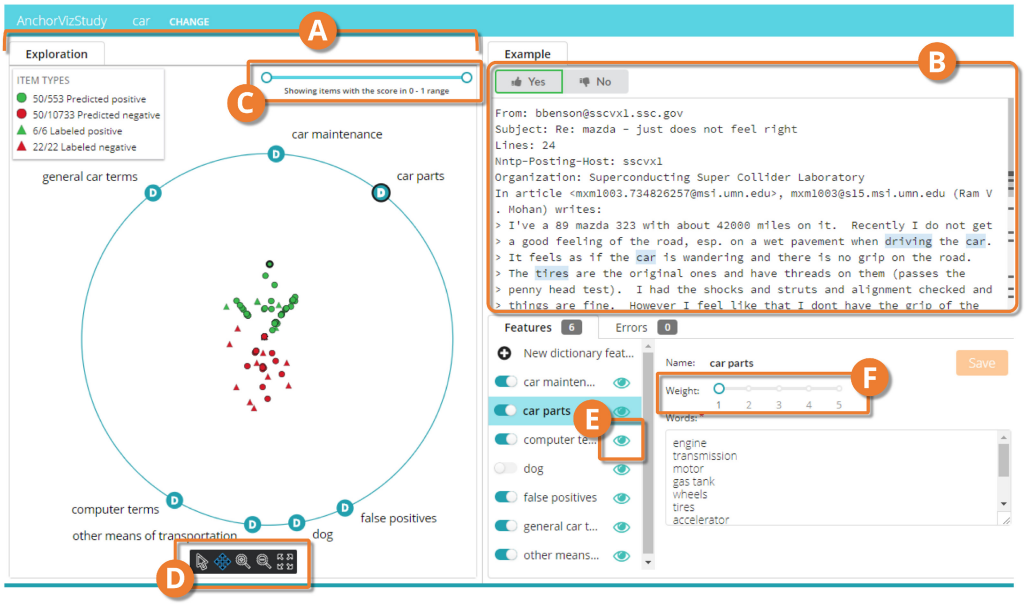

Fig. 8. The Visualization interface contains the Exploration pane on the left that hosts the AnchorViz (A) with a range slider to filter the items based on their prediction scores (C) and a zoom and pan control bar (D). The Example pane (B) displays the content of the selected item. An anchor can be shown or hidden from the visualization with the eye toggle button (E), and its weight can be updated using a slider (F).

that the users can get a quick glimpse of the presence of features in the item (Figure 7(E)). In the Features pane (e.g., Figure 7(F)), the users can create or modify dictionary features by providing a name used to help them organize and manage different concepts (e.g., fruits) and a list of words that are semantically related to each other (e.g., apple, orange, banana). Dictionary features were case-insensitive, but we did not support stemming or lemmatization to avoid users having to understand and potentially debug the stemming algorithm. Features could be turned on or off with a switch to be included or excluded from model. Feature highlights update accordingly to reflect the currently active features. To support the iterative classifier building loop, we trained the classifier on every label or feature activity and returned new samples and predictions according to the latest trained classifier. In addition, we control for the label quality by providing the ground truth labels when the participants submit each label and asking them to correct their label if it is incorrect.

6.2.2 Condition-Specific Interfaces. The main difference between the two conditions was the means of choosing the item to label in the exploration pane.

The Uncertainty condition presents users with an exploration pane (Figure 7(A)), where they get items by selecting "Uncertainty" or "Search" from the exploration drop-down selector (Figure 8(B)). The uncertainty sampling presents one item of which the current classifier is most uncertain (i.e., the prediction score is closest to its decision boundary). Since the Visualization condition allowed text-based search (indirectly through dictionary-based anchors), we provided similar keyword search functionality in the Uncertainty condition.

In the Visualization condition, we use subsampling to visualize top 100 items that the current classifier is most uncertain of on the topology defined by the dictionary-based anchors. Dictionary-based anchors and dictionary features are closely related. When a dictionary-based anchor is created, it can be used as a feature; when a dictionary feature is created, it can be used as an anchor. Each dictionary feature has a toggle *eye* button (Figure 8(E)) for the feature to be active as an anchor in the visualization's ring in addition to the switch to include or exclude the feature in training the model. In other words, the concepts in the Features panel can be used as anchors, or model features, or both.

This second implementation of AnchorViz is informed by feedback from our first study as well as pilot runs. Instead of hierarchical clustering, we chose dataset subsampling because our implementation of clusters was "*more confusing than helpful*" (pilot participant) at cold start. Perhaps low performance and unreliable predictions of the early-stage classifier reduce the value of hierarchical clustering at cold start stage. Evaluating the usefulness of such different strategies for reducing visual clutter remain the subject of future research. We also simplified the types of glyphs, used only two colors to encode both labels and predictions, and used shape to distinguish labeled from unlabeled items. This choice removes the double-encoding from the first implementation and aligns better with the labeling experience. We added the capability to filter the data points based on a range of prediction scores (Figure 8(C)) as well as to change the weight of the anchors (Figure 8(F)) as these functionalities were requested by several participants in the earlier study. Filtering items on prediction scores helps explore items for which the model is confident about a prediction. Changing the weight of an anchor makes an anchor *pull* related items harder, bringing them even closer to it. This feature is useful to resolve occlusions in dense groups of items.

6.2.3 Dataset, Classifier, and Setup. For the second user study, we changed the data type from webpages to a plain text dataset because we observed that the participants were leveraging the structure of the webpage (e.g., embedded images, document structure) which was not available to the learning algorithm to label items even though the tutorial emphasized the classifier's capability to only process the text. This choice helps participants focus on what the learning algorithm

processes and to see if our qualitative observations from the first study can be confirmed using a different dataset.

We used 20 Newsgroups [33], a popular public text dataset consisting of UseNet messages from 1993 belonging to 20 different discussion categories. Of the 20 categories, we chose *cars* and *gun politics* concept modeling as the two study tasks because we determined that these discussion topics remain familiar to the participants. We used *motorcycles* as a target concept for the tutorial task. The sampling data consisted of 11314 messages (594 about cars, 546 about gun politics) and a test data consisted of 7532 messages (396 about cars, 364 about gun politics).

To demonstrate the cold-start experience using the iML tool, we followed a strategy based on anecdotal experiences from expert users of similar iML tools to build an initial classifier. The strategy involved searching the unlabeled dataset for five random positive items that contained the category keyword (e.g., cars, guns) and five random negative items that did not contain the category keyword. In addition to these ten labels, the strategy recommends adding a default feature to kick off the training. To satisfy this requirement, we included a default a *text length* feature which is often useful in text classification tasks. We then trained the cars and guns classifiers, and the resulting classifiers had training accuracy of 50% and 60%, respectively, with the decision threshold of 0.5, making them random or slightly better than random classifiers.

6.2.4 Participants. Recruitment for the second user study was like the first study. We recruited 17 participants from a large software company. We used three participants for the pilot study, and we used data from the remaining 14 participants (3 female) for the analysis. To eliminate the need to teach basic ML concepts during the study, we screened the participants to have basic knowledge about binary classifiers. Three participants have taken some courses in ML (S1-3), three occasionally build ML models in practice (S4-6), five frequently build ML models (S7-11), and three have a doctoral degree in ML-related fields (S12-14). Our participants' professional roles in the company are that of applied data scientist (6), researcher (2), and software engineer (6). None of the participants from the second study were participants of the first study.

6.2.5 Procedure. We conducted this study through video conferencing sessions which we recorded for further analysis. During the sessions, participants shared their screens and thought aloud while performing their tasks. In each session, we introduced the participants to the iML loop and gave a brief tutorial on how to use various components of the iML tool to iteratively build a motorcycles classifier. This onboarding took between 10–15 minutes. The participants then proceeded to building the classifiers for the two conditions. For each condition, we gave a short tutorial for the condition specific exploration pane and explained the topic category; then participants spent 20 minutes building the classifier and completed a short post-task survey about the task they just did. Lastly, they completed their session by filling out a survey comparing the two interfaces (5 minutes). Each session lasted between 70–90 minutes. We compensated each participant with a $30 gift card.

6.3 Quantitative Data Analysis

Our study presented participants with two interfaces designed to help them build a binary classifier. While we normalized most of their functionalities, these interfaces are, by design, not functionally equivalent. The Uncertainty interface (machine-driven approach) provided an affordance for passive exploration while the Visualization interface (human-driven approach) provided an affordance for active exploration. Since the two interfaces are not isomorphically equivalent and our sample size is small ($n = 14$), we did not test for the significance of the treatments. Instead, we explored various ways to quantitatively evaluate the output of the iML sessions, which we present here.

Table 1. Summary Statistics for the Training Set that the Participants Collected during Their 20-minute Sessions

Topic	Interface	Feature Count	Label Count	Label Rate	Test Accuracy	AUCPR
Cars	Uncertainty	8.00 (4.58)	36.29 (12.02)	0.88 (0.12)	0.79 (0.09)	0.86 (0.09)
	Visualization	9.29 (2.29)	25.29 (9.89)	0.80 (0.11)	0.79 (0.07)	0.90 (0.05)
Guns	Uncertainty	6.86 (3.02)	48.29 (31.73)	0.90 (0.09)	0.68 (0.03)	0.77 (0.08)
	Visualization	7.14 (2.61)	27.00 (5.10)	0.79 (0.10)	0.69 (0.08)	0.82 (0.07)

Each value represents the average across participants, and the standard deviation is shown in parentheses. Label rate represents the fraction of viewed items for which the participants submitted labels. AUCPR is the area under the precision-recall curve.

There are several challenges in evaluating iML systems when both labeling and featuring are driven *solely* by humans. It is difficult to design a controlled experiment that can tease apart variability in humans from the design of the system as every human interaction has a compounding influence on the next interaction and eventually the outcome. Furthermore, both the human and the underlying ML model evolve over time. Typical challenges in human-in-the-loop ML, such as label noise, concept evolution, concept drift, feature engineering, motivation and attention, can be studied independently, but generalizing the findings to a realistic human-driven iML scenario that combines all those factors is complex beyond the scope of this article.

Potential metrics to evaluate the output of the session include the ones we used in the first study, but this would be inappropriate for several reasons. The first study only looked at a frozen state of a classifier (i.e., no training after each interaction). Error type analysis is not useful because the learning algorithm can easily overfit to the small set of labels and features. Anchor effectiveness in pulling error items cannot apply when the classifier is at near-random performance.

Despite the above challenges, we present and discuss several methods for evaluating the quality of the training data that the participants generated during their 20-minute, cold-start, interactive classifier building sessions. Our iML sessions allowed for evolving the label and feature set in concert. In other words, the participants added a feature when they discovered concepts from the items, and they searched for items to label when they recalled a concept relevant to the task or identified a related or ambiguous concept from an item. Since the labels and features were created in concert, we evaluate the performances of the classifiers built from the participants' sessions; we also include independent analysis of labels and features as they are typically evaluated in other literature.

6.3.1 Training Set and Session Classifier Performance. On average, participants labeled 34.2 items ($\sigma = 19.2$) and created 7.8 features ($\sigma = 3.2$) with test accuracy of 73.8% ($\sigma = 8.6\%$). While the post-task survey revealed that there is no difference in means of the participants' perception on the familiarity of the topic (cars = 4.2, guns = 4.1; see Table 2 for exact prompt), we saw a significant effect from the topic on the test accuracy ($V = 95$; $p = 0.005$ from paired Wilcoxon signed rank test). Therefore, we present separate results for each topic from this point on. Table 1 shows the summary statistics for the training set and the test performance of the classifiers that they built during their 20-minute sessions.

While the participants created a roughly equal number of features in each interface on average, we observed that the participants using the Uncertainty interface labeled 43.5% and 78.8%, cars and guns respectively, more items than using the Visualization interface; we expected this as the uncertainty sampling presented items for the users to label while AnchorViz required manipulation of anchors and concept topology before choosing an item to label. Despite such differences in the label count, the overall test accuracy of the resulting classifiers is quite similar

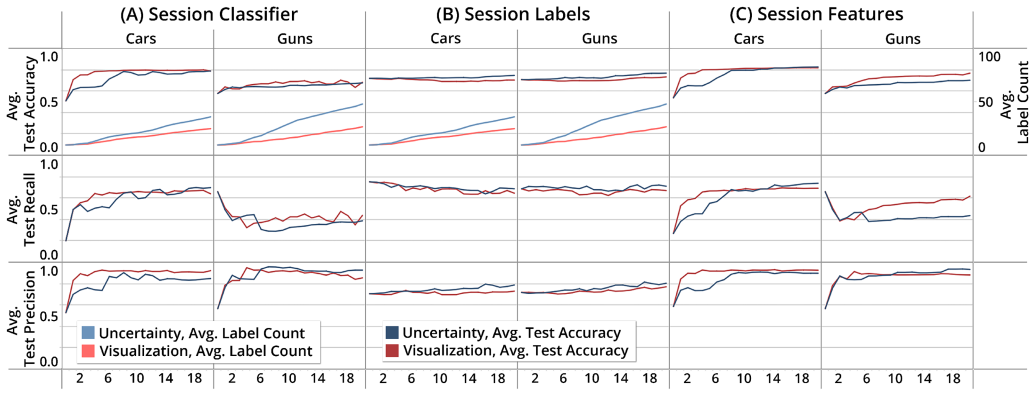

Fig. 9. Average classifier performance over 20-minute sessions for each interface. Average label count is overlaid to provide context for the size of the training set. For each classifier topic, we trained three different classifiers and measured their accuracy, recall, and precision on a held-out test set at every minute elapsed in the participants' sessions. (A) Session Classifier is trained using participants' labels and features, (B) Session Labels is trained using participants' labels and BoW features, and (C) Session Features is trained using participants' features and the full train set. For the cars topic, the classifiers built using the Visualization interface on average arrived at higher test accuracy earlier than the Uncertainty interface. For the guns topic, the Visualization interface outperformed Uncertainty interface for both accuracy and recall.

(see Figure 9(A)). Furthermore, out of all the items that the participants viewed, participants using the Uncertainty interface labeled 9.7% and 14.4%, cars and guns respectively, more items than using the Visualization interface.

Figure 9(A) shows the average test accuracy, recall, and precision of the participants' classifiers over time. For the cars classifier, we see that the Visualization interface arrives at higher test accuracy, recall, and precision earlier that the Uncertainty interface.

6.3.2 Label and Feature Coverage. We observed that the participants viewed *and* labeled fewer items using the Visualization than when using the Uncertainty interface. These observations suggest to us that the Visualization interface helps users to more efficiently extract useful information to build the classifier and to prevent unnecessary labeling.

For our analysis, we defined *label coverage* as a measure of the amount of information that the items provide to help generalize the target concept. We do this instead of looking at *label quality*, which is often used to represent the noise in the labels (i.e., incorrectly labeled items). As a proxy for measuring label coverage, we use a method of evaluating the exploration strategies commonly used in ML, which is to fix the feature set and look at the test performance of the resulting classifier across the number of labels added to the training set. To conduct this analysis, we took the participants' labels over time and trained a BoW classifier. The results are shown in Figure 9(B) along with the average label count. We see that the classifiers built with the Uncertainty interface perform only slightly better (4.4% and 3.6%, cars and guns, respectively) while the number of labels is significantly larger.

Similarly to evaluating the label coverage, we define *feature coverage* as the quality of the feature representations to extract appropriate signals to generalize the target concept. We looked at the feature coverage through the test performance of the classifiers built using participants' features and all the available training set. We trained each classifier using a balanced set of training items (1,188 for cars and 1,092 for guns). The performances of the resulting classifiers are shown in Figure 9(C). For the cars topic, we observe that the classifiers built with the participants' features

using the Visualization interface arrives at a higher test accuracy and recall faster than the Uncertainty interface. This observation agrees with what we presented earlier for the session classifiers (Figure 9(A)). For the guns topic, the classifiers built with the Visualization interface outperforms the Uncertainty interface by 6.7% in average test accuracy and 18.0% in average test recall while maintaining similar precision.

6.4 Qualitative Analysis of Participant Exploration Strategies

As with the first study, we qualitatively coded the usage behavior through recordings of our participants' conversation and usage of the tools. We focus our analysis on feature or concept discovery just as our first study focused on feature blindness discovery. From this analysis, we extracted three main user activities and present how each interface supported these activities during the study: (1) discovering new concepts, (2) labeling concepts, and (3) debugging and refining concepts. These concept activities, while they seem heavily focused on featuring, have strong relationships to exploration and labeling. This is because exploration, labeling, and featuring are tightly coupled together in an iML loop.

6.4.1 Discovering New Concepts. For our analysis, a new concept is any new dictionary feature that the participant creates. Most participants discovered new concepts from items they inspected. Only one participant (S12) recalled a concept from memory ("DMV" for cars classifier) without requiring an item as a potential source of ideation. There were two ways to discovering new concepts. Either the item contained brand new information undiscovered before, or the familiar concept within an item helped participants recall new concepts (e.g., S7 recognized "founding fathers" and recalled "second amendment" for the guns classifier).

In the Uncertainty interface, the participants were able to go through many uncertain items to discover errors or new concepts. This, however, required inspecting each item carefully to determine relevance or quickly scanning and missing important concepts. Searching for keywords required a starting point which some participants perceived to be challenging ("*I don't know what I'd wanna search for.*" - S7). The Visualization interface had a similar challenge regarding search, but the participants readily leveraged the existing anchors to filter out items that are already known and inspect items unaffected by anchors at the center. However, occlusion of data points at the center meant that the participants were not able to go through many items. Some participants (S3, S8, S12) created anchors from the center items, regardless of its relevance to the target concept, to pull some of these items out for inspection. We also observed that, when participants discovered foreign language items, neither interface provided an affordance for dealing with these items at all. This was a tradeoff between the example-based and dictionary-based anchors which we had predicted in the first study.

6.4.2 Labeling Concepts. For our analysis, concept labeling occurs when a user's mental association of the concept to the positive or negative class for the target concept takes place (e.g., "types of cars" is a positive concept to a cars classifier). Concept labeling is like feature labeling except that these labeled concepts are not necessarily used as features for the classifier until the user chooses to do so. Neither interface allowed explicit labeling of concepts other than through naming, but the Visualization interface allowed interactions with these labeled concepts.

Concept labeling is pivotal to AnchorViz, and all participants leveraged concept labeling in the Visualization interface to explore the data. The most common use of the anchors was to look for predictions that are inconsistent with the user's expectation based on the concept represented by the anchor (Figure 10(A)). In other words, the participants looked for predicted negative items along positively correlated anchors (e.g., "types of cars" anchor for a cars classifier) or predicted positive items along negatively correlated anchors (e.g., "motorcycles" anchor for a cars classifier).

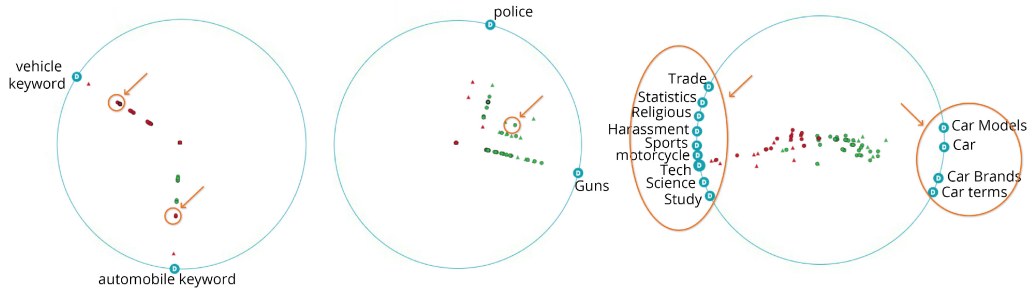

Fig. 10. Three different participants' strategies. Left = Concept labeling allowed S14 to find potential prediction errors along positive concept anchors. Middle = S9 explored the *grey area* between a potentially confusing negative anchor and a positive anchor. Right = S3 put all negative anchors on the left and all positive anchors on the right to achieve separation of predictions.

In certain cases, the concepts were explicitly labeled by their associated class. S5, S12, and S13 named their concepts explicitly as "negative," and S11 named his concept as "false positive." While most anchors were positively or negatively correlated to the target class, some participants labeled potentially ambiguous or confusing concepts as "grey areas"; S12 created an anchor explicitly named "grey areas."

Another powerful use case for concept labeling in AnchorViz is the manipulation of a concept topology. In most cases, participants put positively associated anchors on one side and negatively associated anchors on the other (Figure 10(C)). In other cases, they grouped anchors based on their semantic relationship to each other. This flexibility in customizing a concept topology allows for defining unique concept spaces. For example, S9 called the space between two anchors as "grey area" (Figure 10(B)), while other participants explored the spaces between anchors to find interesting items. On the other hand, the Uncertainty interface did not allow for any organization of the concepts to exploit such a visual concept layout.

6.4.3 Debugging and Refining Concepts. For our analysis, we defined concept debugging as an activity where users inspect the relationship between the items and a concept, or the relationship among two or more concepts regardless of its level of specificity (e.g., "AK-47" is specific while "gun politics" is broad). In the Uncertainty interface, the uncertainty sampling did not provide any means to debug concepts, but the keyword search functionality did.

Concept debugging in the Uncertainty interface was *example-driven*: participants searched for items containing a keyword to confirm whether the items were predicted correctly by the current classifier. This style of debugging made participants inspect each item and look at existing feature highlights to understand the feature influence on the item. Ten participants (71.4%) used search functionality during exploration.

On the other hand, AnchorViz provided an at-a-glance, *distribution-driven* approach to concept debugging. Seeing the distribution of predictions along the specific anchor allowed participants to determine if the concept or feature had a precision problem (by looking at predicted positive items along a negative concept anchor) or a recall problem (by looking at predicted negative items along a positive concept anchor). Participants would then decide if the concept needed to be debugged or refined ("*A lot of examples with 'self protection' terms are negative. That's kind of interesting.*" - S8) or to move onto another concept to debug ("*Things are seemingly well separated.*" - S11).

Thirteen participants (92.9%) moved anchors around to see the predictions of items that followed ("*Oh my gosh, this is not what I was expecting. What's going on here?*" - S13). Seven participants (50%)

Table 2. Participants' Survey Responses for Each Interface on Their Agreement to the Statement (5 = Strongly Agree)

Statement	uncertainty	visual
I am familiar with the topic that I was building a classifier for.	4.2	4.1
It was easy for me to find good examples to label.	4.1	3.9
It was easy for me to identify good features to add.	4.1	3.7
It was easy for me to fix errors.	3.6	3.9
It was easy for me to understand how my classifier was doing.	3.6	3.9
I find this interface useful for building a classifier.	4.0	4.2
Overall, I am satisfied with the interface.	3.8	4.0

changed the weight of the anchor, and six participants (42.9%) turned off unnecessary anchors to strengthen the focus on a specific concept.

In addition, AnchorViz allowed the participants to understand how a concept correlates or confuses with another. Some participants (S8, S9, S12, S14) inspected items between two anchors to discover and refine the grey area, and S12 leveraged the prior knowledge of the probability of co-occurrence of two concepts to debug a concept ("*Certain locations have more gun violence than others. I'm using it to find edge cases, not necessarily as a feature*").

6.5 Participant Feedback and Preference

Table 2 outlines the participants' evaluation of the two interfaces from the post-task survey where there were no significant differences between the two interfaces. In the post-task survey, the participants were asked to enumerate at most three things they most liked and three things they most disliked about each interface, and in the post-study survey, the participants commented on their preference for the interface. We grouped these open-ended responses into (1) general feedback about the iML tool, (2) feedback specific to Visualization interface, and (3) feedback specific to Uncertainty interface. We then produced an affinity diagram for each of these groups.

The three most liked general functionalities of the tool were interactive featuring, feature highlights, and examining the items along with the predictions. Specifically around interactive featuring, the participants liked that features could be hand-crafted and easily updated with instant retraining of the classifiers. The near real-time update of the feature highlights and the classifier predictions provide the necessary feedback to validate their actions. There were several requests for advanced functionalities on featuring, such as feature metrics (e.g., feature importance, frequency, weight), stemming and stop word removal, splitting of dictionaries, and feature suggestions. Some participants wanted summary statistics to determine if the classifier is "ready to go" (S12), and others were concerned that the tool would not scale to meet the needs of big data problems that require thousands of classifiers.

Most participants thought the Uncertainty interface was simple and intuitive, and some liked the "uncertainty prompting" (S4) for labeling efficiency and for exploring ambiguous range (S2). On the other hand, one participant commented that uncertain items were not useful and "felt like random examples" (S12). Search was the most liked functionality because it allowed them to search for specific terms in the domain. However, participants expressed the need for supporting advanced feature debugging activities which was available in the Visualization interface. For example, S8, who started with the Uncertainty interface, requested additional feature information ("*If I add a new feature, how many things does that feature alone misclassify or … how many pos[itive] and neg[ative] examples does it show up in?*"). After he was presented with the Visualization interface, he "*liked that the feature distribution is somewhat visible with the anchors.*" This was also

confirmed by S10 who exclaimed, as soon as he saw the Visualization interface after the Uncertainty interface, that "*this is what I meant before, this is effectively visualizing feature importance!*" S7 also commented that the Uncertainty interface was "*more opaque how a feature differentiates the data points*" after having exposure to the Visualization interface earlier.

Most participants felt that the Visualization interface helped give more insight into how the classifier was doing, helped debug errors, and helped understand the impact of their features by visualizing their interaction with the data. Some participants liked the interactivity and control over the visualization and the visual representation of the classifier and its predictions while others pointed out several shortcomings of the visualization design such as visual clutter and occlusions. One suggestion was to present only items affected by the anchors to avoid cluttering at the center of the visualization. The spatial layout of the concepts using the magnet metaphor helped participants prioritize confusing items and separate and group data. The push/pull metaphor was intuitive but required some getting used to. Some participants also had a hard time figuring out where to begin or getting unstuck. For example, S11 commented that he "*need[ed] more help to get out of a local optimum where everything seems to be working right,*" and wished for a way to "*shake up model behavior . . . with rapid bursts of labeling via [the] uncertainty interface.*" Similarly, S9 wanted a way to select a whole region from the visualization and "*open a batch of items in [the] uncertainty view.*"

In the end, 8 out of 14 participants (57.1%) preferred the Visualization interface, four participants (28.6%) could not prefer one or the other, two participants (14.3%) preferred the Uncertainty interface, and none chose neither. Since the two interfaces provided different affordances, it is no surprise that S2 and S14 wanted the combination of the two interfaces. In addition, the qualitative feedback from the likes and dislikes reaffirm the complementary nature of the two interfaces; the Uncertainty interface allows for quick labeling while the Visualization interface allows for feature debugging.

7 DISCUSSION AND FUTURE WORK

Our two user studies provide an initial exploration into the efficacy of visual semantic exploration and concept discovery in iML, and we discuss several future research directions based on our findings.

7.1 Evaluation of IML Systems

The most common way of evaluating ML systems, interactive or not, is through looking at the summary statistics of the resulting model classifier (e.g., accuracy, F1 score, AUC, etc.). However, in iML systems, these metrics obscure the human interactions and insights, which have traditionally been notions difficult to quantify. In evaluating AnchorViz and the classifiers that our participants built, we faced the challenge of telling the story of our participants' use of the tool through statistics and discuss these challenges here.

7.1.1 Evaluating User Artifacts and Intermediary Steps. In our studies, we carefully controlled for the label noise or concept evolution (i.e., "the labeler's process of defining and refining a concept in their minds [30]") of the target class by focusing on the selected items rather than the labels that the participants provided. Given a fixed definition of the target class, what are appropriate ways to evaluate the tangible artifacts (e.g., features, labels) and intangible artifacts (e.g., insights, mental models, steps) that the user creates during the iML process?

Our goal was to evaluate the extent to which AnchorViz helped users explore and extract diverse and useful information for the target classifier, and we defined and measured label coverage. Another way of looking at the label coverage could involve fully deconstructing the unlabeled dataset

into its conceptual constituents (through automatic, manual, or hybrid clustering techniques) and measuring how much of the structure the selected items cover. Both cases above ignore what portion of the item the user processed to determine its label, so using the entirety of the item is an approximation for what the user *discovered*.

We can consider features as the explicit encoding and externalization of this discovered information, and we measured the effectiveness of the interface in helping users encode the discovered information into anchors and features through feature coverage. But what about the steps taken to discover these features? In AnchorViz, anchors are used as intermediary tools to discover new items and features or to define these features once refined and debugged. We captured the effectiveness of the anchors through AEP and AER metrics. However, these metrics depend on the current classifier's performance, and therefore, do not characterize the anchors independently. For example, an anchor could be entirely cohesive or intelligible to the users but could be ineffective in highlighting the prediction errors because the classifier already has the appropriate representation for that concept. But a cohesive or intelligible anchor can be reused as a contrast to another anchor (i.e., by defining a space of grey areas) or reused in building a different classifier altogether. Anchors or concepts represent knowledge extracted from the dataset and is beneficial to the overall model building experience regardless of its direct impact on the model's performance.

Similar considerations between cohesiveness, intelligibility, and effectiveness exist for features. Feature importance or correlation may depend on the current classifier's performance while feature cohesiveness or intelligibility may not. A feature could precisely capture the users' concept but have no effect in improving the classifier performance. A feature of a hundred thousand dimensions (e.g., BoW) could produce a highly accurate classifier but no human could grok what the feature represents. Feature labeling or concept labeling is another intermediate step that the participants took for exploration of the data and refinement of the classifier. Naively, looking at the correlation between the concept and the labels or the predictions would uncover effective uses of the concept as anchors or features. However, we have also observed that a concept does not need to be correlated with the target concept class for exploration (e.g., grey areas).

We explored the definitions of label and feature coverage and observed intermediary steps and artifacts (e.g., concept labeling, anchors) that common ML metrics do not capture. In trying to evaluate the benefits of AnchorViz, we have uncovered the need for generalizable metrics for evaluating iML systems supporting semantic models, and we encourage a more comprehensive look at defining such metrics.

7.1.2 Productivity and Engagement. Our second, within-subject study was designed to contrast two distinct approaches for finding good items to label: a machine-driven approach (uncertainty sampling) and a human-driven approach (AnchorViz). Most participants preferred the Visualization interface over the Uncertainty interface, and our qualitative analysis of the participants' comments revealed interesting trade-offs and opportunities for the design of iML systems.

Participants liked the Uncertainty interface because they felt that the interface was simple and intuitive for novice users, and the mechanics of labeling the items presented to them felt more "*efficient because of lack of choice* - S7." We observed that some participants preferred to take the *backseat* and let the perceived intelligence drive the exploration. Some felt more productive because they were able to label a lot more items faster. S1 mentioned that, "*[I] felt like more work is being done because it's showing me intelligent items one example after another.*" S7 lamented about her *lack of productivity* with the Visualization interface ("*I only ended up labeling about like … 10 more examples!*") even though she built the best performing cars classifier (test accuracy of 88.5%) with just 23 labeled items (10 given + 13 submitted) and 10 features with only 20 minutes of usage. This observation brings forth a popular misconception in ML that more labels are better; this

may be true for systems with high capacity (i.e., complexity of relationships or parameters that the algorithm can model). In an interactive setting where features are added incrementally, our result indicates that a higher number of labels does not necessarily lead to better classifier performance. Indeed, choosing features carefully and a few labels to exploit them may be the most efficient approach.

Participants liked the Visualization interface because they felt that they were given "*more choice over which examples to label* (S10)." Some participants thought that the Visualization interface was more fun ("*This is actually really fun*" - S7) and engaging ("*It's very engaging because the sample set changes with every iteration.*" - S11). While the sense of control and interactivity provided a fun and engaging experience, it also added a learning curve to the users in terms of understanding how to use the visualization as well as knowing what to do next.

The appropriate balancing of productivity, control, and engagement are important motivating factors in building a ML model, especially when all activities within an iML loop (selecting, labeling, featuring, debugging) require human attention and interaction. Our post-task or study survey did not explicitly measure the participants' perception of productivity or engagement, and it is unclear how such subjective metrics can be measured accurately. We see this as an opportunity for further research in improving the overall iML experience.

7.2 Sensemaking

In the human-in-the-loop ML systems, we often assume that humans are oracles that possess the knowledge, the knowledge transfer is unidirectional from humans to the ML algorithm, and the ML system needs to efficiently learn from the knowledge provided by the humans. However, our scenarios of a food blogger trying to discover unknown unknowns or a student trying to build a gun conversation classifier from scratch require an iML tool that *closes the loop*. In other words, the human in the iML loop not only provides the available knowledge but also iteratively gains an understanding of the dataset and the classifier. The human learning is then transformed into a set of actions and artifacts (i.e., labels and features) that changes the state of both the human and the machine. This iterative process of information gathering, hypothesis forming and testing, refining and re-evaluating the mental representation, and presenting the findings is best known as a *sensemaking* loop [52].

Based on our observations from the two user studies on the interactions and user behaviors, we believe that AnchorViz supports sensemaking in iML. Our users are able to search the dataset for an interesting item (i.e., one that contradicts their expectation), formulate a hypothesis about the specific item (e.g., there should be a lot of recipe index pages) or the concept discovered in that item (e.g., the word "accident" can be a confusing feature), collect evidence for and against the item or concept through the use of anchors, and act on the conclusion they derive from the evidence (i.e., add or modify a label or feature). Table 3 illustrates how AnchorViz facilitates the sensemaking loop using Pirolli and Card's model [48]. Making sense of a large dataset using ML and visual analytics is a growing research area [10, 16], and we propose that a special focus and future research is necessary for scenarios specific to iML where human-driven concept exploration takes place.

7.3 Better Exploration Strategies

During our first study, we observed that participants who interacted with more contrasted items had better performance gains on the test set, that contrasted items have a higher chance of being unknown unknown errors, and finally that some participants were able to discover rare concepts (e.g., edible flowers, cocktail recipes). During our second study, we observed that some participants were selecting seemingly random items on the visualization, and some participants were only

Table 3. Application of Pirolli and Card's Sensemaking Loop on AnchorViz with Ten Processes and Six Representations

Bottom-up Processes	
Search & filter	Search using example-based or dictionary-based anchors; Sort based on the distance from the anchors and filter based on contradictions or agreements between predictions and labels (of the items and the concepts)
Read & extract	Read the items and label them; Inspect the items to identify whether such item belongs to the target concept; Extract concepts relevant (or unrelated) to the target concept or the current hypothesis
Schematize	Create a new example-based anchor from the item that could help represent the concept; Create a new dictionary-based anchor from keywords that define a concept; Add items or keywords to existing anchors; Organize the anchors by providing appropriate names, forming relations between the anchors and the target class, and moving them to the appropriate location
Build case	Inspect the items around and along the anchors of interest to validate the anchors' effect on the data
Tell story	Submit the labeled items or features to the trainer
Top-down Processes	
Reevaluate	Trained model provides new prediction scores; Features used by the model is highlighted
Search for support	Determine if the retrained model removes the visual inconsistencies
Search for evidence	Inspect the items inside the anchors or around the anchors to determine if the retrained model correctly learned the new concept
Search for relations	Identify other related concepts within the inspected items
Search for information	Create a new example-based or dictionary-based anchors based on the newly discovered concept; Navigate up and down the cluster stack to find other interesting patterns given new anchors; Organize and move the new anchors to find other visual inconsistencies
Representations	
External data source	Unlabeled and labeled dataset; Existing features and anchors
Shoebox	Selected set of items that represent a newly discovered concept
Evidence file	Labeled items; Keywords that represent a concept
Schema	Newly created anchors or modified anchors and their spatial layout
Hypotheses	"Current classifier is blind to a (newly discovered) concept."; "Current classifier is already knowledgeable of a concept."
Presentation	Labeled items; Features representing the new concepts

looking for items that did not match the expected prediction along the anchor dimension. Many participants expressed the need for best practices or tips on how to use the visualization because of the tool's learning curve. From our observation of how participants use the visualization, we extracted several effective strategies. They are:

- Focus on one specific concept at a time and limit the use of anchors to only those that matter.
- Give meaningful names to anchors.
- Focus on the contrasts both in position along the anchor dimensions and colors.
- Place anchors at semantically meaningful positions such as contrasted anchors at 90 degrees or at opposite sides.
- Encode all discovered concepts (positive, negative, ambiguous, or unrelated) as anchors to rule them out from potential new concepts to discover.

- Take advantage of advanced anchor operations such as changing the weights of an anchor, combining anchors, or contrasting anchors.

We believe that controlled studies on the utility and effectiveness of these strategies are important and should be the subject of future work.

8 CONCLUSION

We have introduced AnchorViz, a novel interactive visualization that facilitates semantic data exploration and concept discovery for iML. The need to discover potential blind spots and select good examples for building an ML model motivated this visualization, and we evaluated it through two user studies that demonstrate one of the few efforts in looking at the fully human-driven iML loop. In our first study, we observed that semantic visual exploration using AnchorViz helped users discover feature blindness errors, which are sources of ideation for improving the underlying classifier. In the second study, we observed that integrating AnchorViz in an end-to-end iML loop helped users build a classifier that achieves equivalent or better performance than a similar system relying on uncertainty sampling, while using fewer labels and producing higher quality features at the early stages of the model building process.

In both cases, AnchorViz provided a visual language that enabled users to interact with labeled and unlabeled data to both learn and teach new concepts. The extracted knowledge which takes the form of new concepts or new features was neither explicitly in the data (which is unlabeled) nor in the humans (e.g., surprise discovery of insect and flower recipes). The power of the visualization remarkably brought these new concepts to the foreground. Uncertainty sampling, by sampling around 0.5 scores, is optimized to maximize the information of each label when no prior information is available from the human. We expected it to be difficult to beat the uncertainty sampling given that, even though the human has additional information (other than just label), we do not know *a priori* how to extract it. However, interacting with AnchorViz was comparable to and even outperformed uncertainty sampling in some cases.

We currently support defining an anchor through the information in existing items in the dataset, or by dictionaries of semantically related keywords. Existing items typically contain multiple concepts and are not organized for the users to systematic isolate useful sub-concepts. Dictionaries have the opposite limitation and do not model higher level concepts. Both these sources of information support different level of granularity or cohesiveness in concepts. Future work should investigate expanding the anchor language (e.g., allowing existing semantic models to become anchors) to allow for the systematic and progressive exploration of the *unknown space*.

REFERENCES

[1] Charu C. Aggarwal and ChengXiang Zhai. 2012. *Mining Text Data*. Springer Science & Business Media.

[2] Jae-wook Ahn and Peter Brusilovsky. 2009. Adaptive visualization of search results: Bringing user models to visual analytics. *Information Visualization* 8, 3 (2009), 167–179.

[3] 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.

[4] Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 337–346.

[5] 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.

[6] Joshua Attenberg, Panos Ipeirotis, and Foster Provost. 2015. Beat the machine: Challenging humans to find a predictive model's "unknown unknowns". *Journal of Data and Information Quality (JDIQ)* 6, 1 (2015), 1.

[7] Josh Attenberg and Foster Provost. 2010. Why label when you can search?: Alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 423–432.

[8] Michael Brooks, Saleema Amershi, Bongshin Lee, Steven M. Drucker, Ashish Kapoor, and Patrice Simard. 2015. FeatureInsight: Visual support for error-driven feature ideation in text classification. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST'15)*. IEEE, 105–112.

[9] Mackinlay Card. 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan-Kaufmann.

[10] Duen Horng Chau, Aniket Kittur, Jason I. Hong, and Christos Faloutsos. 2011. Apolo: Making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 167–176.

[11] Nan-Chen Chen, Jina Suh, Johan Verwey, Gonzalo Ramos, Steven Drucker, and Patrice Simard. 2018. AnchorViz: Facilitating classifier error discovery through interactive semantic data exploration. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces*. ACM, 269–280.

[12] Justin Cheng and Michael S. Bernstein. 2015. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 600–611.

[13] Jason Chuang, Sonal Gupta, Christopher Manning, and Jeffrey Heer. 2013. Topic model diagnostics: Assessing domain relevance via topical alignment. In *Proceedings of the International Conference on Machine Learning*. 612–620.

[14] Aron Culotta, Trausti Kristjansson, Andrew McCallum, and Paul Viola. 2006. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence* 170, 14–15 (2006), 1101–1122.

[15] Pedro Domingos. 2012. A few useful things to know about machine learning. *Commun. ACM* 55, 10 (2012), 78–87.

[16] A. Endert, W. Ribarsky, C. Turkay, B. L. Wong, Ian Nabney, I. Díaz Blanco, and F. Rossi. 2017. The state of the art in integrating machine learning into visual analytics. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 458–486.

[17] Jerry Alan Fails and Dan R. Olsen, Jr. 2003. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI'03)*. ACM, New York, 39–45. DOI: https://doi.org/10.1145/604045.604056

[18] 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.

[19] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and Experience* 21, 11 (1991), 1129–1164.

[20] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 145–154.

[21] Florian Heimerl, Charles Jochim, Steffen Koch, and Thomas Ertl. 2012. FeatureForge: A novel tool for visually supported feature engineering and corpus revision. In *COLING*.

[22] Patrick E Hoffman. [n.d.]. *Table Visualizations: A Formal Model and Its Applications*. Ph.D. Dissertation. University of Massachusetts. Lowell.

[23] Rong Hu, Sarah Jane Delany, and Brian Mac Namee. 2010. EGAL: Exploration guided active learning for TCBR. In *Proceedings of the International Conference on Case-Based Reasoning*. Springer, 156–170.

[24] Xinran Hu, Lauren Bradel, Dipayan Maiti, Leanna House, and Chris North. 2013. Semantics of directly manipulating spatializations. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2052–2059.

[25] Camille Jandot, Patrice Simard, Max Chickering, David Grangier, and Jina Suh. 2016. Interactive semantic featuring for text classification. *arXiv preprint arXiv:1606.07545* (2016).

[26] Ian Jolliffe. 2011. Principal component analysis. In *International Encyclopedia of Statistical Science*. Springer, 1094–1096.

[27] Hannah Kim, Jaegul Choo, Haesun Park, and Alex Endert. 2016. InterAxis: Steering scatterplot axes via observation-level interaction. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 131–140.

[28] Josua Krause, Adam Perer, and Enrico Bertini. 2014. INFUSE: Interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 1614–1623.

[29] Josua Walter Hugo Krause. 2018. *Using Visual Analytics to Explain Black-Box Machine Learning*. Ph.D. Dissertation. New York University Tandon School of Engineering.

[30] 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.

[31] Todd Kulesza, Simone Stumpf, Weng-Keen Wong, Margaret M. Burnett, Stephen Perona, Andrew Ko, and Ian Oberst. 2011. Why-oriented end-user debugging of naive Bayes text classification. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 1, 1 (2011), 2.

[32] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Eric Horvitz. 2017. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. In *Proceedings of AAAI*. 2124–2132.

[33] Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, Vol. 10. 331–339.

[34] Hanseung Lee, Jaeyeon Kihm, Jaegul Choo, John Stasko, and Haesun Park. 2012. iVisClustering: An interactive visual document clustering via topic modeling. *Computer Graphics Forum* 31, 3pt3 (June 2012), 1155–1164. DOI: https://doi.org/10.1111/j.1467-8659.2012.03108.x 00041.

[35] David D. Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*. 148–156.

[36] David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag New York, Inc., 3–12.

[37] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. 2015. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2017), 1249–1268.

[38] Shixia Liu, Xiting Wang, Mengchen Liu, and Jun Zhu. 2017. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics* 1, 1 (2017), 48–56.

[39] Shixia Liu, Jiannan Xiao, Junlin Liu, Xiting Wang, Jing Wu, and Jun Zhu. 2018. Visual diagnosis of tree boosting methods. *IEEE Transactions on Visualization & Computer Graphics* 1 (2018), 1–1.

[40] Yafeng Lu, Rolando Garcia, Brett Hansen, Michael Gleicher, and Ross Maciejewski. 2017. The state-of-the-art in predictive visual analytics. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 539–562.

[41] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[42] Brian Mac Namee, Rong Hu, and Sarah Jane Delany. 2010. Inside the selection box: Visualising active learning selection strategies. In *Proceedings of The Challenges of Data Visualization Neural Information Processing Systems (NIPS) Workshop*. Dublin Institute of Technology.

[43] Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

[44] Christopher Meek. 2016. A characterization of prediction errors. *CoRR* abs/1611.05955 (2016). http://arxiv.org/abs/1611.05955

[45] Gregory Murphy. 2004. *The Big Book of Concepts*. MIT Press.

[46] R. M. Nosofsky. 1986. Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology. General* 115 1 (1986), 39–61.

[47] Kai A. Olsen, James G. Williams, Kenneth M. Sochats, and Stephen C. Hirtle. 1992. Ideation through visualization: The VIBE system. *Multimedia Review* 3 (1992), 48–48.

[48] Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of the International Conference on Intelligence Analysis*, Vol. 5. 2–4.

[49] Hema Raghavan, Omid Madani, and Rosie Jones. 2005. InterActive feature selection. In *Proceedings of IJCAI*, Vol. 5. 841–846.

[50] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. 2017. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan. 2017), 61–70. DOI: https://doi.org/10.1109/TVCG.2016.2598828 00000.

[51] Eleanor Rosch and Carolyn B. Mervis. 1975. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology* 7, 4 (1975), 573–605.

[52] Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. 1993. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems (CHI'93)*. ACM, New York, 269–276. DOI: https://doi.org/10.1145/169059.169209

[53] John W. Sammon. 1969. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers* 100, 5 (1969), 401–409.

[54] Sam Scott and Stan Matwin. 1999. Feature engineering for text classification. In *Proceedings of ICML*, Vol. 99. 379–388.

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

[56] Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.

[57] John Sharko, Georges Grinstein, and Kenneth A Marx. 2008. Vectorized radviz and its application to multiple cluster datasets. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008).

[58] Ben Shneiderman. 1992. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.* 11, 1 (Jan. 1992), 92–99. DOI: https://doi.org/10.1145/102377.115768

[59] Edward E. Smith and Douglas L. Medin. 1981. *Categories and Concepts*. Vol. 9. Harvard University Press, Cambridge, MA.

[60] Robert R. Sokal. 1958. A statistical method for evaluating systematic relationship. *University of Kansas Science Bulletin* 28 (1958), 1409–1438.

[61] Ji Soo Yi, Rachel Melton, John Stasko, and Julie A. Jacko. 2005. Dust & magnet: Multivariate information visualization using a magnet metaphor. *Information Visualization* 4, 4 (2005), 239–256.

[62] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies* 67, 8 (2009), 639–662.

[63] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics* 37, 2 (2011), 267–307.

[64] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S. Tan. 2009. EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, 1283–1292. DOI: https://doi.org/10.1145/1518701.1518895 00097.

[65] Stef Van Den Elzen and Jarke J van Wijk. 2011. Baobabview: Interactive construction and analysis of decision trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST'11)*. IEEE, 151–160.

[66] Alfredo Vellido Alcacena, José David Martín, Fabrice Rossi, and Paulo J. G. Lisboa. 2011. Seeing is believing: The importance of visualization in real-world machine learning applications. In *Proceedings of the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2011: Bruges, Belgium, April 27-28-29, 2011*. 219–226.

[67] Byron C. Wallace, Kevin Small, Carla E. Brodley, Joseph Lau, and Thomas A. Trikalinos. 2012. Deploying an interactive machine learning system in an evidence-based practice center: abstrackr. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. ACM, 819–824.

[68] Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H. Witten. 2001. Interactive machine learning: Letting users build classifiers. *International Journal of Human-Computer Studies* 55, 3 (2001), 281–292.

[69] Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*, Vol. 97. 412–420.

[70] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert. 2018. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* (2018).

Received May 2018; revised September 2018; accepted November 2018