AnchorViz: Facilitating Classifier Error Discovery through Interactive Semantic Data Exploration

Nan-Chen Chen[1], **Jina Suh**[2], **Johan Verwey**[2], **Gonzalo Ramos**[2], **Steven Drucker**[2], **Patrice Simard**[2]
[1] University of Washington, Seattle, WA, USA
[2] Microsoft Research, Redmond, WA, USA
nanchen@uw.edu, {jinsuh, joverwey, goramos, sdrucker, patrice}@microsoft.com

ABSTRACT

When building a classifier in interactive machine learning, 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 paper presents *AnchorViz*, an interactive visualization that facilitates error discovery through semantic data exploration. By creating example-based anchors, users create a topology to spread data based on their similarity to the anchors and examine the inconsistencies between data points that are semantically related. The results from our user study show that AnchorViz helps users discover more prediction errors than stratified random and uncertainty sampling methods.

Author Keywords

interactive machine learning; visualization; error discovery; semantic data exploration; unlabeled data

INTRODUCTION

Interactive machine learning (iML) is a growing field in machine learning (ML) that emphasizes building models with humans in the loop. Unlike traditional ML's sequential development workflows, model developers in iML iteratively explore and label data, add features to fix errors, and modify models to enhance performance. As models constantly get feedback from humans, their evolving directions can align with the model developers' goals better. Since data are labeled on the fly in iML, there is no way to tell how the current model performs on the unlabeled set until these data points (hereafter, we refer to a data point as an "item") are labeled. As it is undesirable to label the whole dataset in iML, it becomes critical to efficiently locate unlabeled items that the current model will predict incorrectly (i.e., *errors in unlabeled items*).

One common approach in iML 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 errors, often the selected items are ambiguous or random errors. Errors with high prediction confidence are not likely to be selected by these sampling algorithms. Hence, model developers may not even be aware of such defects in their models. Furthermore, these sampling methods do not fully leverage the value of human-in-the-loop. They treat humans as oracles and cause humans to feel annoyed, lose track of teaching progress, or lose interest [2]. As the discovered errors are mostly uncertain or random, it can be challenging for humans to make sense of the errors to provide useful inputs.

Past discussions have pointed out the benefits of semantic models – models that have semantically meaningful feature representations [11]. Semantic models are useful in helping humans make sense of the model behaviors and ensure their predictions are not based on problematic associations. 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 challenging to the model, and try to fix the errors by providing more labeled items or adding features. However, it is costly and unrealistic to ask humans to provide an exhaustive list of concepts and items, especially for rare cases that are hard to recall. Therefore, careful design is required to efficiently utilize human efforts and maximize the value of each interaction.

In this work, we extend the idea of building semantic models to include semantic exploration on unlabeled dataset for error discovery and propose a polar-coordinate-based interactive visualization, *AnchorViz*, to facilitate this semantic exploration process. Through the visualization, users can create example-based anchors to spread items based on pair-wise similarity. Since users can understand the meaning of the chosen items, they can associate and contrast items' relations to the anchors. Hence, they can tease apart concepts and find items whose predictions do not align with the corresponding concepts.

We evaluated AnchorViz with 16 participants to examine the users' exploration strategies and the use of anchors in the tool in building a binary classifier. Our results show that items highlighted through anchors are more likely to be errors for which the current classifier has no corresponding features (hereafter referred to as "feature blindness errors"). In addition, our participants used a variety of strategies to explore the dataset and distill positive and negative concepts of the target class through creating anchors.

IUI'18, March 07–11, 2018, Tokyo, Japan

ISBN 978-1-4503-4945-1/18/03…$15.00

DOI: https://doi.org/10.1145/3172944.3172950

The contributions of this work are three-fold: First, we highlight the opportunity for leveraging semantic data exploration to locate classifier errors in unlabeled items. Second, we design and build AnchorViz to support such exploration through interactive visualization. Third, we evaluate AnchorViz with users and analyze users' intentions and exploration strategies to show that AnchorViz opens a new research space for semantic interactions with an iML classifier.

Motivating Scenario

Discovering an error in a large dataset is like finding a needle in a haystack, and in some cases, one cannot even tell whether a needle is in the haystack. Sifting through the haystack haphazardly is an inefficient method to find the needle. Using a magnet to pull the needle would be a fine solution. However, one needs many different magnets to find all kinds of needles, each representing a missing concept in the model. In addition, as these needles are unknown, it can be tough to create corresponding magnets to pull them out.

In this work, we leverage this magnet analogy, but instead of creating magnets that aim to pull needles directly, we propose to use semantic magnets, called "anchors" in this paper, to decompose the dataset semantically and highlight prediction inconsistency in items that are neighbors in the semantic space. In other words, anchors are representations of concepts in the dataset. An anchor can *spread* the dataset based on the items' semantic similarity to different concepts. If one expects to see items of a certain type of prediction near an anchor, any item with a different prediction from its neighbors or the nearby anchor can be an error or a seed to discover new concepts.

To illustrate how one can create and use anchors in a classification task, we provide the following scenario: A food blog writer wants to build a binary classifier for finding recipe webpages. The writer already considers restaurant menu pages as negative, so the writer uses existing restaurant menu pages to create a *restaurant menu* anchor to see other items similar to the concept of restaurant menus. Among the items similar to restaurant menu concept, the writer notices that some items are being predicted as positive. Upon inspection of those predicted positive items, the writer discovers catering webpages. The writer considers catering webpages as not belonging to recipes, labels the discovered catering webpages as negative, and creates a *catering* anchor. With both *restaurant menu* and *catering* anchors, the writer can see if there are other similar items being predicted incorrectly.

BACKGROUND AND RELATED WORK

In a supervised iML setting, humans build models by providing training items and features in a quick, iterative and continuous loop [9]. This model building approach has shown its success in building a decent model using fewer features [30]. In addition, it provides meaningful interactions to improve the user's trust and understanding of the system [27]. This interaction between humans and machines goes beyond simply treating humans as label oracles and requires thoughtful design and careful user studies [2]. However, like traditional supervised ML, supervised iML also needs to find items to which the current model is blind for better generalization performance in the real world, which is the focus of this paper.

Unknown Unknowns and Prediction Errors

Attenberg et al. [3] and Lakkaraju et al. [13] 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 are useful for fine-tuning the system. Unknown unknowns are errors for which the model has high confidence, and such errors can be potentially disastrous depending on the problem domain.

Another approach is to characterize prediction errors as feature blindness, ignorance, mislabeling, or learner errors [17]. 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, 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 represents missing concepts in the current model.

Searching for Items to Label

There are two types of strategies for searching items to label: Machine-initiated and human-initiated approaches. The machine-initiated or algorithmic approach uses learning algorithms to suggest items for humans to label so that the model needs fewer training items to perform better [22]. 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 [16, 15]. 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 [3, 13, 17].

Recently, Lakkaraju et al. introduced an algorithmic approach to find unknown unknowns directly [13]. 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 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 undesired consequences of losing interpretability [11]. In our work, we aim to preserve semantic meanings during exploration to maximize interpretability.

The human-initiated approach allows the humans to find the items that the model should learn. Attenberg et al. introduced the notion of *guided learning* [4] and implemented *Beat the Machine* (BTM), where crowd workers are tasked to find items from the open world [3]. Their results showed that BTM found more unknown unknowns compared to a stratified random sampler. Guided learning puts a significant load on users 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) [25] 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.

Visualization for ML

Visualization is one key approach to facilitating model development. Prior work that supports specific ML tasks usually contains a form of interactive visualization. For example, Talbot et al. designed *EsembleMatrix* for creating and tuning ensemble classifiers [28]. Lee et al. visualized topic models and allowed users to modify the models through their interface [14]. Ren et al. used a parallel coordinate style of visualization to help examine classification errors [21]. However, existing work mostly focuses on discovering errors in the labeled set with little attention to finding errors in unlabeled items.

In AnchorViz, we focus on locating errors in unlabeled items by using visualization to spread out items in a semantic manner. Our design is inspired by VIBE [19] and Dust and Magnet (D&M) [26]. D&M uses magnet metaphor to attract similar items using pre-defined dimensions of the data. However, the existing dimensions in our iML scenario (i.e., features) may not have any connections with unlabeled items to *attract* them. Similar to Adaptive VIBE [1], we allow the users to create and refine "anchors" as our version of magnets. We explain the anchors in further details in the next section.

Semantic Memory and Concept Decomposition

In psychology, semantic memory is a type of human memory that stores general knowledge, such as facts and concepts [29, 20]. Network models are commonly used to describe semantic memory; the theory of hierarchical network models [5] points out that a concept can be decomposed into smaller concepts to store in memory. In ML, concept decomposition is a learning method that leverages clustering [6, 7, 8]; the goal is to divide a concept into smaller concepts so that the algorithm can learn easily. Our work is inspired by these two similar ideas, and we put focus on semantic exploration and decomposition of the dataset through anchors.

ANCHORVIZ

In this section, we introduce the design of *AnchorViz*, an interactive visualization to help model developers locate classifier errors. Our design is based on the following design objectives that we came up with based on the motivating scenario:

DO1. Let users define concepts of the target class and unrelated classes

DO2. Spread the dataset based on concepts

DO3. Show how user-defined concepts impact the positions of items

DO4. Provide information about the model's current prediction along the user's labels

DO5. Optimize for efficient reviewing process

Interface and System Design

The interface for AnchorViz (Figure 1) contains a RadViz-based visualization that shows both labeled and unlabeled items (Figure 1C) as well as anchors that represent concepts (Figure 1D). When users click an item, the thumbnail view (1B) will switch to show its content, and users can inspect the item to provide a label. This gives users a quick way to check if the item is an error and provide correct label (DO5).

Define Concepts with Example-Based Anchors

In this work, we let users define a concept via examples (DO1), which is a common view of how a concept is stored in brain [18]. Thus, these anchors are "example-based" anchors. We leave exploring other types of anchors as future work. To create an anchor, users can drag an item to the outer circle (Figure 1D). Users can also drag an item to an existing anchor to modify the corresponding concept. When a user clicks on an anchor to reveal the anchor detail view, 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. In addition, users can hide an anchor from the visualization. The hidden anchors sit inside the "Anchor Repository" at the bottom drawer of the left pane (Figure 1F) where users can restore the anchors to the visualization.

Manipulate Topology and Layout

Once a concept is defined, the user should be able to see the correlation between the concept and the items (DO2; e.g., this webpage is more like "catering") as well as the relationship between several concepts with respect to the items (DO3; e.g., this webpage is more like "catering" and less like "restaurant menu" while that webpage is unrelated to both concepts). We map the relative similarity of the items to the concepts to the position of items in a non-orthogonal coordinate system in a circle; the center point of the outer circle to each anchor forms a set of axes on a 2D surface. 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 (Eq 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 in the anchor. The final position of an item in the visualization is the sum of the vectors to each anchor (Eq 2).

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

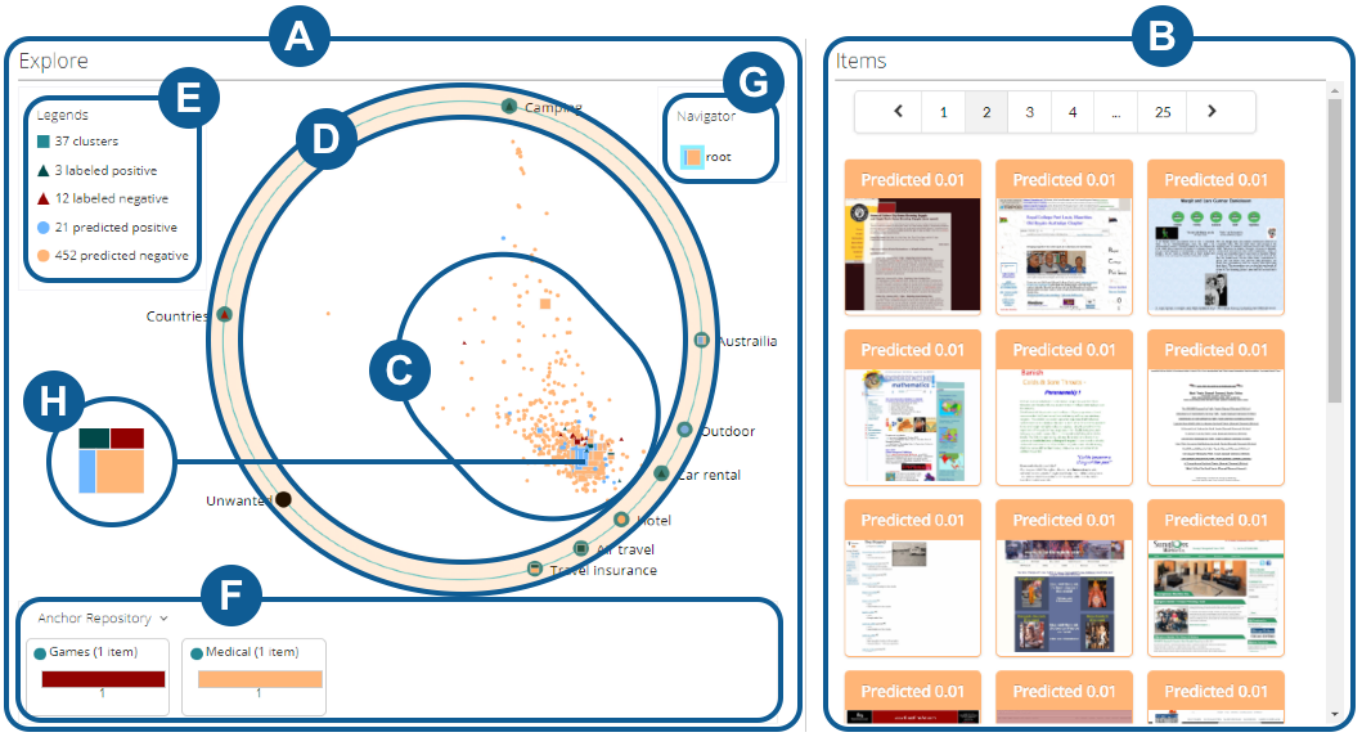

Figure 1. Overview of AnchorViz interface. The interface has Explore pane (A) that includes the visualization and Items pane (B) which shows a paginated grid of thumbnails of all currently visible items in the left pane. The visualization shows data points (C) within the outer circle (D) where anchors are positioned. The legend for data points (E) also acts as filters. Anchor repository (F) contains unused anchors. The navigator (G) shows which cluster the visualization is displaying in the current navigation stack. Clusters are represented as treemap-style squares (H).

Thus, the items that are closer to an anchor are more similar to the items in the anchor. The items that are affected by moving the anchor will move along with it, whereas items that do not share any similarity will remain still (DO3). In addition, since we want to ensure all items sit within the outer circle, an item's value on an axis is normalized by the sum of all its values on all the axes. This normalization follows the typical normalization used in RadViz [10]. In this way, the users are effectively creating a topology in the semantic space defined by the anchors. We have considered multiple visualization techniques for defining this topology, but ultimately chose RadViz because of its support for arbitrary number of axes and its flexibility in positioning of axes while preserving the relative independence of the axes [23]; we leave other visualization options for future work.

Contrast Model Predictions with Concepts

Semantic data exploration alone does not address our goal of finding errors of a classifier if the model is not involved in the exploration process. By surfacing the model predictions and the labels on the semantic topology created by anchors, the user should be able to contrast the predictions with concepts (DO4) to look for potential inconsistencies in two different ways. First, the user can look for items that are predicted to be in the opposite class as the class expected to be close to an anchor. 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 a sea of negatively predicted or labeled items, that item is an outlier worth inspecting. We encode model predictions and user labels into color and shape of items as illustrated in Figure 1E. Users can also click on the categories in the legend to filter items based on their types.

Inspect Groups of Items

Manipulation of the concept topology defined by anchors and efficient visual encoding of model predictions still leaves the user with hundreds of items to inspect in order to arrive at an error worth labeling and adding to the training set. We pre-process the dataset by grouping similar items using HC algorithm; this helps reduce the search space by limiting the number of visible data points and allowing users to review groups of similar items rather than individual items. Note that the clusters do not change when anchors vary since the goal is to group similar semantic items, not merely visually close points.

In our clustering algorithm, the distance between any two items is their cosine similarity in the BoW space. The distance between clusters (C_a and C_b), including single point clusters, is the average distance between pairs among two clusters (Eq 3).

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

In each step of HC, the algorithm selects a pair 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 HC, which is a binary tree, into a n-ary tree from the top tree node and expand the tree in a breath-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.

These pre-processed groups of similar items (hereafter, we refer to a group of similar items as a *cluster*), are displayed as treemap-style squares (Figure 1H) in the visualization, and the size of the square is a function of the number of items inside the cluster. Namely, each square is also a single-level treemap [24] representing the composition of items in four categories (labeled positive, labeled negative, predicted positive and predicted negative). With treemaps, 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.

Each square representing a cluster is positioned in the visualization based on the average similarity of all the leaf items inside the cluster. Clusters move along with the anchors that they are most similar to just like any individual items. When users click on a cluster, they can navigate into the cluster and the visualization will update to show the items and clusters belonging to the cluster. The navigator on the top right (Figure 1F) shows which cluster the visualization is displaying in the current navigation stack. The "root" cluster at the top of the stack represents the entire dataset. Users can navigate back to a previous level through the navigator or double click on the white space in the Explore pane.

Implementation

Our system has 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, are persisted to disk on the web server to support browser refreshes and for evaluation of the study results.

EVALUATION

We conducted a user study to evaluate our visualization and its effectiveness in helping users 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.

User Study

Participants

We recruited 20 participants from a large software company. Initially, we designed a controlled experiment comparing the visualization to an uncertainty sampler for effectiveness in discovering errors. However, the pilot study with the first four participants revealed that learning the tool and the target class took significant amount of time. Since our goal is not to test the ease of learning of the tool, we changed our study design to evaluate the visualization in depth. Our analysis is only based on the remaining 16 participants (7 females/9 males, ages 25-55) who completed the study.

We categorized our participants into four groups according to their ML background. Four participants have taken 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). Six participants have a doctoral or professional degree, nine have a master's degree, and one has a bachelor's degree.

Background Scenario and Tasks

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 cannot recall 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) 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.

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 paper.

To control for the ratio between the labeled set and the unlabeled set, we uniformly sampled for 4000 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 in order 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. 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.

Procedure

We conducted our study through video conferencing where participants shared their screens and thought aloud during the study, and we audio and screen-recorded the entire session. The user study consisted of four parts: The first part (20 minutes) 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 (10-20 minutes) 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 is used for the actual task. The third part was the actual task (20 minutes) where we asked the participants to use AnchorViz to find and label a diverse set of items where the participants disagree with the classifier's predictions. At the end of the study (5+ minutes), we asked participants to complete a quick survey to assess their satisfaction and collect open-ended feedback. Each study was a 70-90 minute session with a $30 gift card compensation.

Quantitative and Qualitative Data Analysis

We focused our analysis on the items that the participants discovered and the circumstances and the behaviors surrounding their discovery. These items should represent concepts which the model is blind to and the users could not think about in isolation. We captured the participants' interaction and behavior through recordings of their conversation and usage of the tools, and we instrumented the web application for user behavior to replay their interactions with anchors or items.

Qualitative Coding

We used the recordings in depth to understand the behavior of the participants. Since the participants followed the think aloud protocol, we were able to replay the recording to interpret the context of their actions. 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.

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 are feature blindness errors by individually retraining the classifier with every item and see if that item is 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, there is not a single item with reference to "insect recipes" 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.

Anchor Effectiveness

To measure the effective 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 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.

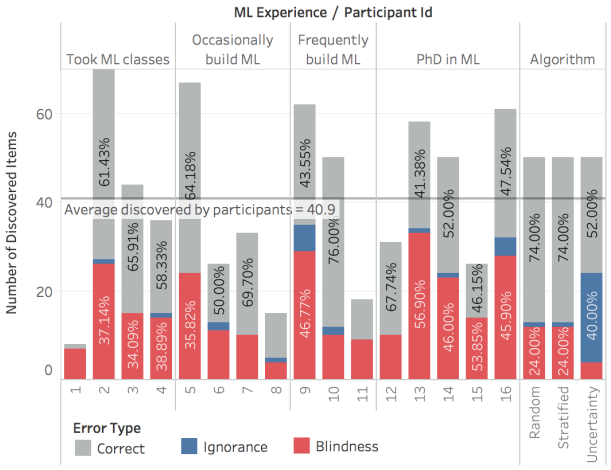

Figure 2. Distribution of errors across participants and algorithmic samplers. Text in each bar shows the percentage of errors in different categories. All participants except for P10 discovered feature blindness errors at a rate higher than any algorithmic samplers.

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:

$$AEP = \frac{\text{\# true errors in contrasted items}}{\text{\# contrasted items}} \quad (4)$$

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

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 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.

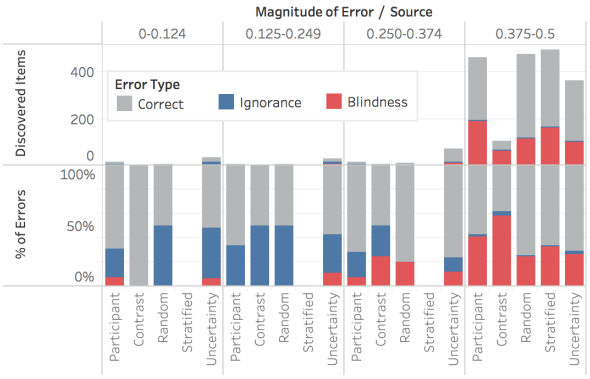

Figure 3. Distribution of discovered items (top) and the magnitude of errors (bottom) across participants and algorithmic samplers. The items are skewed towards high magnitude errors, but items discovered by participants and contrasted items higher chance of getting high magnitude items than algorithmic samplers.

RESULTS

Discovered Items

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).

Error Analysis

On average, participants discovered 40.9 errors (*SD*=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 for PhD in ML group) 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 2 shows the distribution of errors among discovered items across participants.

Figure 3 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 among the discovered (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 participants and contrasted group had a higher chance of discovering errors than algorithmic samplers. As expected, the uncertainty sampler selects more ignorance errors than any other samplers.

Test Accuracy

Assuming that the features were 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

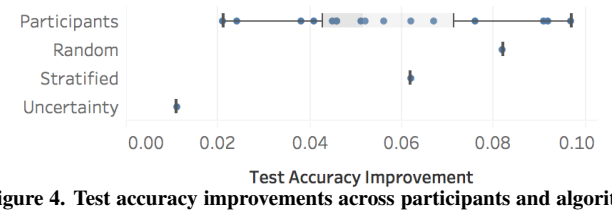

Figure 4. Test accuracy improvements across participants and algorithmic samplers. All participants built better performing classifiers than uncertainty sampler, and three participants built better performing classifiers than random samplers. While random samplers performed well on the test set, they discover less feature blindness errors as indicated by Figure 3

the training set. As before, we selected first 50 items from algorithmic samplers for comparison. Figure 4 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 less feature blindness errors as indicated by Figure 3, and three of our participants (P9, P13, P16) had classifiers with higher test accuracy than the ones created by random samplers.

Anchor Effectiveness

Figure 5 shows the average AEP and AER of each participant. The reference line "Random" of AEP shows the baseline for AEP to be 0.255, 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 ($= 0.1321$) 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 the error analysis section.

User Behavior

In this section, 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.

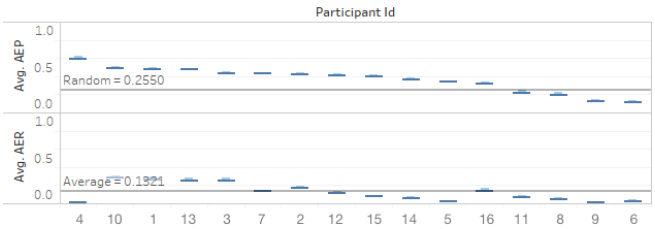

Figure 5. Average anchor effectiveness metrics (AEP/AER) for each participant. Most participants have better AEP than random.

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].*"

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 in the opposite class from the labels of the items in the anchors.

As participants are interacting 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.

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 webpages that contain 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.*"

DISCUSSION AND FUTURE WORK

Concept Evolution and Feature Discovery

Our evaluation reveals that AnchorViz not only helps users to find prediction errors, but also supports them in performing other necessary classifier building tasks. In particular, we observed that the participants used the anchors to facilitate concept evolution, which is "the labeler's process of defining and refining a concept in their minds [12]." Several participants used the anchors to collect items that they were unsure about (e.g., cookbooks, recipe links), deferred the decision for class assignment, and bulked label the items at a later point. 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, as P2 commented that this interface is useful for "*finding features I didn't know about and finding instance that can represent some useful concepts.*" P13 said, "*it was also really useful for discovering items that were out of scope (e.g., foreign language).*"

Anchor Types and Manipulation

In our current design, we explore the use of example items and determine their similarity in the BoW space. However, creating an anchor with example items is only one way to define its corresponding concept. We envision to use other types of anchors and distance metrics in AnchorViz. For example, instead of creating anchors based on items that users encounter, the tool can support search to enable direct access to specific items, and create an anchor based on all items with the given search terms. This is similar to P10's suggestion: "*keyword search through the data to find tougher items from intuition and see where they lie on the interface.*" In addition, the tool can let users specify a set of keywords to compute prevalence of the keywords or allow defining any arbitrary function that outputs a normalized score given any item. Furthermore, the current design treats all anchors equally. It is possible to add weights to anchors so that users can express their levels of interests on different concepts. We leave all these directions for future research.

Integration into Interactive Model Building Loop

A typical iML loop involves selection and labeling of items, featuring, and training the model in an iterative loop. The evaluation in this paper is focused on the selection step of finding errors that are worth labeling because selecting right examples is critical to efficient model building process. Though we leave the evaluation of AnchorViz in iterative iML loop for future works, we discuss below how such a tool can be incorporated into all stages of iML.

Once an error is discovered through the tool, labeled and added to the training set, a new model reclassifies the dataset; the user could continue to select the next item if there are no additional training errors or address new training errors by modifying features or adding new labeled items. For example, a positive label on a webpage about an edible flower recipe could be classified as negative even after retraining a model if the model does not have the appropriate feature representation for it (i.e., feature blindness error). The system would prompt the user to provide a feature to fix the error, and upon adding an "edible flower" feature, the model is finally retrained to correctly classify the webpage as positive. In addition, the tool could be useful for bulk labeling of items within an anchor or items co-located in a topology created by many anchors. Since the tool supports capturing concepts in the form of anchors, another use of the tool is for the evaluation of the model performance at the concept level by looking at the error rate within the anchor or around the anchor.

Some participants also wanted support for feature debugging. During the study, participants were making best guesses as to why the classifier was making a mistake (P16 - "*I think there are few words. That's why it's not able to classify it as recipe.*"). Providing feedback about the model's current feature set and additional anchor types could remedy this issue. For example, the user could create an anchor representing the number of tokens in an item to see if it is indeed true that low token count correlates with negative prediction.

Furthermore, the anchors contain rich information that could be used for suggesting useful features or highlighting the discrepancies among an anchor's items in the current model's feature space. After any model-changing action (featuring or labeling), a new model is trained to reclassify the entire dataset for the user to evaluate whether the action led to an expected outcome (e.g., better performance, discovery of new errors).

Interface Improvements

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!*"

However, many improvements could be made: P7 wanted to see a "*succinct summary*" of the items instead of thumbnails when she was looking at items around the anchor to see if there were any obvious errors. Several participants wanted the ability to select multiple items for bulk labeling or adding to anchors. Several participants wanted to overlay additional data into the visualization. For example, a user would search for keywords and see the highlighted search results against the current anchor configuration. Instead of using the predicted label, actual prediction score could be used as color. In addition to the current filters, participants wanted to filter based on a range of scores such that they could focus on items around the decision boundary or with high prediction confidence.

Further Evaluation

So far, we have evaluated the interface for a binary classification scenario and a fixed dataset. Further investigation into how the visualization will be used in different contexts is necessary. While we observed that the visualization can be useful for general data exploration, it would be helpful to understand when to promote the interface to people who are less familiar with ML through evaluating the visualization at different model building stages (i.e., cold start, ready to deploy) and with models of different performance characteristics (i.e., recall, precision, error distribution). Varying the distribution of positive items or labeled items in the exploration set could also potentially change the effectiveness of the visualization. One participant suggested that this tool could be useful for exploring an image dataset with model features as anchors; another wished to extend the visualization for multiclass scenarios. There is an opportunity to extend the application of the visualization to different ML problems or data types.

Finding a baseline tool to compare against is a challenge due to many confounding and complex variables such as choice of samplers, choice of workflow, and variability in the user's abilities (e.g., the ability to feature and debug the model). Nevertheless, further investigation into comparing the visualization against a baseline is necessary to quantify the benefits of tool.

CONCLUSION

This paper presents AnchorViz, an interactive visualization tool for semantic data exploration and error discovery in iML. Our user study shows that AnchorViz helps users discover more prediction errors than stratified random sampling and uncertainty sampling. We have enumerated various exploration strategies used by the participants and discovered that the participants used AnchorViz beyond its intended usage with potentials for facilitating concept evolution and feature discovery. AnchorViz can be extended to support features, models, and active learning algorithms which opens several possibilities for future research.

ACKNOWLEDGEMENT

We acknowledge all the reviewers for their valuable feedback and the participants for their time, as well as the Machine Teaching group at Microsoft Research for their support.

REFERENCES

1. 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.

2. 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.
3. 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.
4. Josh Attenberg and Foster Provost. 2010. Why label when you can search?: alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 423–432.
5. Allan M Collins and M Ross Quillian. 1969. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior* 8, 2 (1969), 240–247.
6. Inderjit S Dhillon and Dharmendra S Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine learning* 42, 1 (2001), 143–175.
7. Inderjit Singh Dhillon and Dharmendra Shantilal Modha. 2003. Concept decomposition using clustering. (May 6 2003). US Patent 6,560,597.
8. Jasminka Dobsa and BJ Basic. 2003. Concept decomposition by fuzzy k-means algorithm. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*. IEEE, 684–688.
9. 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, NY, USA, 39–45. DOI: http://dx.doi.org/10.1145/604045.604056
10. Patrick E Hoffman. *Table visualizations: a formal model and its applications*. Ph.D. Dissertation. University of Massachusetts. Lowell.
11. Camille Jandot, Patrice Simard, Max Chickering, David Grangier, and Jina Suh. 2016. Interactive Semantic Featuring for Text Classification. *arXiv preprint arXiv:1606.07545* (2016).
12. 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.
13. Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Eric Horvitz. 2017. Identifying Unknown Unknowns in the Open World: Representations and Policies for Guided Exploration. In *AAAI*. 2124–2132.
14. 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: http://dx.doi.org/10.1111/j.1467-8659.2012.03108.x 00041.
15. David D Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the eleventh international conference on machine learning*. 148–156.
16. 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.
17. Christopher Meek. 2016. A Characterization of Prediction Errors. *CoRR* abs/1611.05955 (2016). http://arxiv.org/abs/1611.05955
18. R M Nosofsky. 1986. Attention, similarity, and the identification-categorization relationship. *Journal of experimental psychology. General* 115 1 (1986), 39–61.
19. 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.
20. Daniel Reisberg. 2013. *The Oxford handbook of cognitive psychology*. Oxford University Press.
21. 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: http://dx.doi.org/10.1109/TVCG.2016.2598828 00000.
22. Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.
23. 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).
24. Ben Shneiderman. 1992. Tree Visualization with Tree-maps: 2-d Space-filling Approach. *ACM Trans. Graph.* 11, 1 (Jan. 1992), 92–99. DOI: http://dx.doi.org/10.1145/102377.115768
25. Robert R Sokal. 1958. A statistical method for evaluating systematic relationship. *University of Kansas science bulletin* 28 (1958), 1409–1438.
26. 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.
27. 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.

28. 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, NY, USA, 1283–1292. DOI: http://dx.doi.org/10.1145/1518701.1518895 00097.

29. Endel Tulving and others. 1972. Episodic and semantic memory. *Organization of memory* 1 (1972), 381–403.

30. 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.