Interactive Machine Teaching: a human-centered approach to building machine-learned models

Gonzalo Ramos[1], Christopher Meek[1], Patrice Simard[2], Jina Suh[1], Soroush Ghorashi[2]

[1]Microsoft Research, [2]Microsoft

ABSTRACT

Modern systems can augment people's capabilities by using machine-learned models to surface intelligent behaviors. Unfortunately, building these models remains challenging and beyond the reach of non-machine learning experts. We describe interactive machine teaching (IMT) and its potential to simplify the creation of machine-learned models. One of the key characteristics of IMT is its iterative process in which the human-in-the-loop takes the role of a teacher teaching a machine how to perform a task. We explore alternative learning theories as potential theoretical foundations for IMT, the intrinsic human capabilities related to teaching, and how IMT systems might leverage them. We argue that IMT processes that enable people to leverage these capabilities have a variety of benefits, including making machine learning methods accessible to subject-matter experts and the creation of semantic and debuggable machine learning (ML) models. We present an integrated teaching environment (ITE) that embodies principles from IMT, and use it as a design probe to observe how non-ML experts do IMT and as the basis of a system that helps us study how to guide teachers. We explore and highlight the benefits and challenges of IMT systems. We conclude by outlining six research challenges to advance the field of IMT.

CONTENTS

1. INTRODUCTION

2. RELATED WORK
   2.1. Lowering the barrier to machine-learned models
   2.2. Leveraging human knowledge
   2.3. Leveraging human capabilities
   2.4. Understanding machine learners and models

3. INTERACTIVE MACHINE TEACHING
   3.1. Model-building goal
   3.2. People as teachers
   3.3. The iterative machine teaching process

4. LEVERAGING AND SUPPORTING HUMAN CAPABILITIES
   4.1. Through a constructivist lens
   4.2. Human capabilities and teaching
   4.3. Supporting human capabilities

5. INSIGHTS FROM AN INTEGRATED TEACHING ENVIRONMENT
   5.1. PICL: a platform for interactive concept learning
   5.2. A design probe of interactive machine teaching
   5.3. A study on guiding novice teachers

6. RESEARCH OUTLOOK
   6.1. Leveraging rich human knowledge and capabilities
   6.2. Teaching languages
   6.3. End-user machine learning and user experience design
   6.4. Machine-in-the-loop teaching
   6.5. Interactive machine teaching literacy and teaching teachers
   6.6. Interactive machine teaching as programming

7. CONCLUSION

REFERENCES

1. INTRODUCTION

Emerging software systems are augmenting and enabling people by surfacing intelligent behaviors driven by machine-learned models. They keep us safe by warning the car driver from inadvertently getting out of a lane, by detecting anomalous financial activities on our credit card, or by

detecting malignant tumor cells from a microscopic image. They also make us productive by augmenting our ability to process, find and organize information by tagging and classifying personal documents, by externalizing our preferences in recommendation systems that learn about what we like, or by amplifying our senses by enabling us to see in low lighting conditions. The potential for machine-learned models to enable intelligent system behavior and augment people's abilities has led to an increased demand for machine-learned models. Unfortunately, this increased demand coexists with inherent challenges associated with building machine-learned models, such as understanding the types of problems that can be solved using ML and navigating the fair and ethical use of ML in building systems. One of the key challenges, and the focus of this article, is that building machine-learned models (ML models) is difficult and requires machine learning (ML) expertise often not possessed by subject-matter experts (e.g., a doctor who can diagnose diseases may not know how to build an ML model that can perform that task). This difficulty makes the process of creating ML models inaccessible to many people.

Human-computer interaction (HCI) researchers have actively pursued these challenges. Research targeting the challenges associated with building ML models by leveraging humans-in-the-loop can be found under the broader topic of *interactive machine learning* (IML). An IML process is one in which a person (or people) engages with a learning algorithm, in an interactive loop, to generate useful artifacts[1]. In some scenarios, these artifacts are data (e.g., the output of a music recommendation system might be a list of songs personalized based on previous interactions of a user). In other scenarios, these artifacts are insights about data (e.g., detecting irregularities in a household's credit card transactions). The focus of this article is the scenarios where these generated artifacts are machine-learned models. These models are functions of the context and user's interaction (e.g., a function that takes an array of strings from a user-specified document as input and outputs a calendar event).

Researchers studying IML have considered a wide variety of different roles for the humans engaged in the interactive loop with the machine learner. These include the roles of ML experts, data scientists, crowdsource workers, and domain experts. Logically, the form and function of IML systems are, in large part, determined by the nature of these roles. Focusing on specific roles is useful because they a) provide specificity and focus for research investigations, b) point to opportunities to leverage role-specific capabilities, and c) provide the vocabulary to engage with users. The focus on this article is on interactive machine teaching (IMT), an IML process in which the human-in-the-loop takes the role of a teacher, and their goal is to create a machine-learned model[2]. Our focus is guided by the potential IMT has to leverage intrinsic human capabilities related to teaching, make ML model-building accessible to subject-matter experts, and lead to semantic and debuggable, easy to maintain ML models.

Our article has the following structure. We first provide context to our research by discussing research work in the areas related to IMT, including IML research (Section 2.). We note that it is not our intention to provide a comprehensive survey of the IML literature but rather to underscore

[1]For a more fine-grained exploration of the design space of interactive machine learning systems see (Amershi, 2011).

[2]We choose the name *interactive machine teaching* to be analogous to IML. It describes the role of the human-in-the-loop and captures the interactive nature of the engagement. As compared with the use of *machine teaching* (MT) as defined by (Zhu, 2015; Zhu, Singla, Zilles, & Rafferty, 2018) the inclusion of the word *interactive* highlights both the human interactive nature of the process and builds on the concept of machine teaching presented by (Simard et al., 2017). We do, however, acknowledge that this leaves some ambiguity: is the machine the teacher or the student? Unlike our interpretation where humans teach to machines, some other works have used the term IMT in the context of a machine teaching humans (Johns, Mac Aodha, & Brostow, 2015).

IML research as it relates to IMT. In the remainder of the article, we present three contributions. The first contribution of this article is to name, frame, and articulate the benefits of interactive machine teaching. We highlight its potential to address many of the challenges faced today in building machine-learned models, as well as outlining several opportunities at the intersection of human-computer interaction and artificial intelligence (Section 3.). Part of this contribution includes a discussion and exploration of how IMT can leverage human capabilities and human knowledge beyond labels to make the process of creating ML models more human-centered and efficient (Section 4.). The second contribution of this article consists of the observations and insights we obtained through a design probe. Our design probe used a prototype *integrated teaching environment* (ITE) aimed at creating machine-learned models for classification and extraction from text documents. This prototype implements several IMT principles and ideas described in previous sections. We present insights from the design probe about the design and implementation of future ITEs. Our third contribution leverages our framing of IMT and our observations of teachers interacting with our ITE, and consists of a set of six research directions to advance the field of interactive machine teaching (Section 6.). We conclude by summarizing our main contributions and highlighting the potential of IMT.

2. RELATED WORK

We look at prior work as it relates to fundamental goals and aspects of IMT and provides examples that (1) aim to lower the barrier for subject-matter experts to create ML models, (2) encourage people to produce human knowledge beyond just labels, (3) leverage and supports inherent human capabilities, and (4) provide information about the machine learner and learned-model so that teachers can evaluate and affect the learning process outcome.

2.1. Lowering the barrier to machine-learned models

Enabling users from different backgrounds and with different expertise to use and apply ML artifacts remains an important goal of multiple research communities. In a traditional ML process, an end-user with ML expertise is often responsible for several tasks: acquire a labeled set of examples, select a testing set, choose features to consider, select an appropriate learning algorithm and its parameters, train the model, evaluate the model's predictions against the test set, and, depending on the results, go back to the beginning. Numerous systems aim at making the process above accessible to non-ML experts by automating stages like algorithm selection, parameter selection, feature selection, and testing. In these approaches known as AutoML, users are mostly limited to providing labeled data. AutoML seeks to automate, and thus removing a user's decision-making process in regards to what learning algorithm to try, hyperparameters to select (Hutter, Lücke, & Schmidt-Thieme, 2015; Fusi, Sheth, & Elibol, 2018), or features to choose (Kanter & Veeramachaneni, 2015). While research in this area is active (Olson, Urbanowicz, et al., 2016; Olson, Bartley, Urbanowicz, & Moore, 2016), the AutoML approach may still not be a solution many end-users prefer as it produces black-box models that are inadequate for scenarios requiring transparency. AutoML can also be impractical to use in cases where there are not significant labeled data available.

Some IML research aims at making ML model building more accessible. An IML process differs from the traditional way of training an ML model, in which a learning algorithm uses a large

number of labeled examples in a single or small number of batches. Instead, IML is "an interaction paradigm in which a user or user group iteratively builds and refines a mathematical model to describe a concept through iterative cycles of input and review"(Dudley & Kristensson, 2018). Work in this area refers to the people engaged in this paradigm as human(s)-in-the-loop (Fails & Olsen, 2003), in particular when they are engaged in a "rapid, focused and incremental learning cycles" (Amershi, Cakmak, Knox, & Kulesza, 2014). This notion of rapid stimulus-response interactions is being revised by the collective desired to use deep models that promising high accuracy yet take longer times to train. Providing a survey of IML fall outside of the scope of this article. We refer readers interested in that topic to read (Dudley & Kristensson, 2018) and (Suh et al., 2019)'s related work section.

From a systems perspective, there are several approaches aimed at making ML accessible for end-users. These approaches are as varied as the types of end-users (e.g., developers, data scientists, designers) and the prior knowledge they bring. ML model creation frameworks are aimed at users that are comfortable writing code. These frameworks often manifest as libraries or modules that a programmer can import into the logic of larger software projects. Examples of these frameworks include scikit-learn (Pedregosa et al., 2011) python library, turicreate (Apple, 2020), or SpaCy (Spacy, 2020). Jupyter (Jupyter, 2020) or IPython (Pérez & Granger, 2007) notebooks provide an environment that supports drafting solutions that leverage existing ML model creation frameworks. Other approaches are model on the notion of an Integrated Development Environments (IDE) that provide end-to-end ML solutions for users familiar with ML concepts. For example, (Patel et al., 2010) presented an example of a system supporting model-building in an integrated environment that includes both data and coding capabilities.

The above approaches take a deliberate point of view on ML literacy by leveraging both the existing literacy around programming languages and end-user programming environments. It is still unclear if this strategy is effective for subject-matter experts with little to no ML or programming expertise, which is a focus of IMT. Experiments like Google's Teachable Machines (Google, 2020b) try to take the approach of bootstrapping an ML building project using a visual experience requiring no ML expertise. Agency over the produced model, however, requires literacy with ML model parameters, programming languages, and ML frameworks like Tensorflow.

2.2. Leveraging human knowledge

Although a common perception about ML is that (lots of) labels are necessary to create a model, in IML systems, users have agency for tasks and ability to provide rich knowledge beyond providing labeled data such as picking appropriate examples to label, selecting or defining features, as well as choosing algorithms and their parameters. For example, *Reinforcement Learning* (RL) (Sutton & Barto, 2018) is an ML approach where a learner can receive positive or negative feedback about its actions from a human. Through this form of IML, the teacher steers the learner towards the desired behavior. Furthermore, RL enables people to provide demonstrations or critiques (Chernova & Thomaz, 2014) to train systems, agents, or robots. Krening et al.'s work (Krening, 2019; Krening & Feigh, 2018, 2019) aligns with our focus on people without specialized training as teachers of intelligent agents. In particular, her work studies the use of language instruction to produce explanations, demonstrations, and critiques in the context of training RL systems, as well as using human-factor metrics when evaluating systems. The bonsai platform (Bonsai, 2020a) is aimed at developers to develop automation industrial AI solutions. It does this by using machine teach-

ing ideas as an abstraction to think about how to structure the training of RL systems. Through a domain-specific language called inkling, end-users decompose objectives into subtasks that are part of lessons to the learner. This act of decomposition is another example of human knowledge beyond labels. This article looks at applying human knowledge outside the RL domain. In particular, we look at supervised learning scenarios, which are an important set of ML solutions that can benefit non-ML experts.

Other learning systems train models behind the scenes. In email clients with spam classifiers, end-users not only provide labels for emails they see but also express rules that the system uses to label the incoming email as spam automatically. The Snorkel system (Ratner et al., 2019) allows users with programming knowledge to create labeling functions (i.e., rules) to accelerate the creation of labeling sets they can use for building ML models. The above examples illustrate methods for eliciting different human knowledge that can be leveraged to build ML models and applications. In particular, as we describe later in this article, we advocate for users with subject-matter knowledge critical to building the model.

2.3. Leveraging human capabilities

Designers of human-centered systems and experiences are mindful of how to best leverage human perception and cognition. Card et al. exemplify this when they talk about "using vision to think" (Card, Mackinlay, & Shneiderman, 1999). Similarly, systems take advantage of people's capabilities in spatial memory (Scarr, Cockburn, & Gutwin, 2013) as well as being better at recognition than recall (Bates, 1998).

By design, ML visual tools can capitalize on human cognitive and perceptive capabilities for making ML models creation as straightforward as directly manipulating graphical objects on a canvas. Efforts such as Machine Learning for Kids (Lane, 2020) use the Scratch (Resnick et al., 2009) visual programming language to introduce ML concepts and the tools to help children and young adults build simple ML models. Other tools such as Lobe.ai (Matas, Menges, & Beissinger, 2020) abstract the complexity of underlying deep models by presenting a visual, composable network of entities representing the flow of input data and the processes and transformations it goes through to produce an ML model's predictions. This approach is similar to the one used by Azure ML Studio (Azure, 2020).

Another way to leverage human capabilities is to relate to concepts or mental models familiar to people. For example, the Wekinator (Fiebrink & Cook, 2010) is an environment that capitalizes on a straightforward mental model where a person maps input signals to (desired) output behaviors. By demonstrating how the system should behave, Wekinator end-users can create musical instruments, control a game, or serve as input for another process. In another example, Cakmak and Thomaz (Cakmak & Thomaz, 2014) show how to generate more efficient training sets by leveraging human intelligence. In their studies, end-users presented with Teaching Guidance, derived from teaching algorithms or heuristics, produced training sets that were more efficient than selecting training examples on their own.

In the case of IMT systems, we want to leverage people's ability to transfer the knowledge of how to accomplish a task to another person. Furthermore, we seek to support the human capabilities involved in the process of teaching.

2.4. Understanding machine learners and models

An essential part of a teacher-student relationship is that of the teacher being able to discern if a student has learned concepts or how to perform a task. Furthermore, teachers can better help a student if they know why a student did not learn. In the case of IMT, for the teacher to interactively improve an ML model, there has to be a way for the teacher to inspect the model results or state in a form that gives agency for improving it.

This issue is directly related to the growing demand (Goodman & Flaxman, 2017) for explainable (Doshi-Velez et al., 2017) or intelligible models. Making sense of how a model arrived at its prediction is essential, especially in mission-critical, high-stakes domains (e.g., healthcare, criminal justice). Interpretability is sought after to gain trust in the system, to be fair and ethical, to inform human decisions and actions (Lipton, 2018). Explanations serve a key communication role between the model and the human (either a model builder or a consumer of the model predictions). Because of the crucial role that explanations play, model explanations and the languages used to articulate them are emerging as desirable characteristics of modern IML systems, and an essential aspect of IMT systems.

From a growing body of research in Explainable AI (XAI), there are two main approaches for developing explainable machine learning models (Weld & Bansal, 2019). Both of these approaches rely on the features of the model to be semantic. Without semantic features, the model will not be interpretable by people.

One approach uses post-hoc computation or learning techniques to generate explanation models that "approximate" the reasoning of the underlying black-box model (e.g., (Ribeiro, Singh, & Guestrin, 2016; Lundberg & Lee, 2017)). Arguably, such approximation is similar to how humans provide justifications as the precise function of their decision making cannot be expressed or known. However, such efforts to explain the inscrutable models have been criticized for their veracity, the potential for misleading, and inconsistencies (Rudin, 2019; Au, 2018). Furthermore, these explanation models require explanations themselves (Gilpin et al., 2018; Mittelstadt, Russell, & Wachter, 2019). Regardless of the approach, there needs to be a robust way to evaluate interpretability and its impact on the target task (Doshi-Velez & Kim, 2017).

Another approach uses learning algorithms that are "inherently" interpretable such as linear models, rule-based systems, or decision trees. A common criticism for these so-called simpler models is that they are not as accurate as more complicated models (i.e., accuracy and interpretability trade-off)(Lipton, 2018; Rudin, 2019). There have been advances in machine learning algorithms that balance these competing goals (Lakkaraju, Bach, & Leskovec, 2016) and achieve both objectives simultaneously (Ustun & Rudin, 2017; Caruana et al., 2015; Valdes et al., 2016).

Explanations generated from approximating the underlying models are merely informative or indirectly actionable, in that they help a person understand, to a certain degree, the inner workings of the model, suggest potential changes to how you might use the model outputs, but not the model itself. Inherently intelligible explanations, however, can be directly actionable; the human can directly manipulate and alter the model to experiment with the model or to reach their goals. As we discuss later in the article, IMT aligns with the latter, where the inherent intelligibility property of the algorithm affords a particular dialog language for inspecting, understanding and directly manipulating the model.

3. INTERACTIVE MACHINE TEACHING

In this section, we provide details about the characteristics of interactive machine teaching. IMT is distinguished from the more general IML by the specific nature of its goal (model-building) and the specific role of the human-in-the-loop (teaching). Also, IMT has specific types of interactions that are supported during the iterative model-building process.

3.1. Model-building goal

User goals in IML systems fall in two main categories: *task completion*, and *model building*. In solutions tailored for task completion goals, a user interacts with a learning system that creates or updates a model that is used to complete the task. In particular, systems with task completion goals are characterized by the person being knowledgeable about the task they want to complete, but not being a machine learning expert, or caring about the means used to complete their tasks. (Ritter & Basu, 2009) is an example of this type of focus, where they present an IML system at the service of supporting selection tasks in a computer interface. Other IML processes that take the task completion goal include various information gathering and recommendation scenarios, such as music, news, or movies.

The goal in IMT is that of building a machine-learned model to complete future tasks. This goal is shared with the field of ML. As a field, ML focuses on the development of algorithms for learning from data (e.g., labels, user interactions, and demonstrations) and optimizing their parameters. At its core, it aims to extract knowledge from data. For instance, it is common that ML research idealizes the person in an IML system to be an oracle that consistently answers questions asked by a learning system. By doing so, researchers can analyze algorithmic questions under idealized user behavior. This approach has led to advances in the state of the art in IML algorithms. Active learning research, a branch of ML, takes this view on the role of people as oracles in an IML system (Settles, 2012). In this context then, the aim of active learning research is to develop algorithms for dynamically selecting examples to label such that collecting and labeling those examples would lead to a high-quality ML model. One current research area in active learning is the exploration of the potential benefits of leveraging oracles that can provide knowledge beyond class labels (e.g., Kane, Lovett, Moran, & Zhang, 2017; Xu, Zhang, Miller, Singh, & Dubrawski, 2017).

There have been significant efforts to create more effective tools, frameworks, and methodologies to support the goal of building machine-learned models. These tools often seek to leverage a user's existing programmer's mindset and skill. Systems like scikit-learn (Pedregosa et al., 2011), spaCy (Spacy, 2020), and TensorFlow (Google, 2020c) are examples of systems that require not only a programmer's mindset but also a tangible level of ML knowledge. An important property of these systems is that they can help end-users produce functionality that they can later apply to new situations. For instance, the goal might be to create a function that can take a photo as input and return a tagged photo where the set of tags describe the different entities that the system detects within the given photo.

For some systems, characterizing the end goal as task completion versus model building is less clear, such as in the Crayons system (Fails & Olsen, 2003), where a user can interact with the system to separate a foreground object of interest in an image. Through this interaction, this system presents its current best segmentation, and the user marks pixels as foreground or background through brushstrokes on an image. After each user interaction, the system updates the

current segmentation to reflect the user's input and presents the segmentation for further review and corrective input, if needed. The Crayons system underlines how goals are sometimes relative, as it can be viewed from the task completion, (image segmentation for a particular image) or model-building (building a pixel classifier for other images) perspective.

3.2. People as teachers

Systems supporting model-building can engage its users and stakeholders in a variety of roles. They can play the role of providing labels (e.g., crowd-workers), demonstrating (e.g., a robot's trainer), defining training data sets (e.g., data scientists), or selecting a particular learning algorithm (e.g., ML practitioners). It follows that different roles require different skills and varying degrees of ML expertise. The role of the people interacting in IMT is that of a teacher. That is, they are experts on a particular subject matter and have the explicit goal of creating an ML model about it through a teacher-student-type of interaction. These processes are at the core of what we call *interactive machine teaching*. The activity of teaching provides a natural starting point for the human-centered design of IML systems. Teaching is a ubiquitous skill that most people have and use in some form, regardless of subject-matter expertise. Also, user experiences that are based on people's intrinsic ability to teach have the high potential to be widely adopted. The notion of systems where a person acts as a teacher is an early one in IML (Ware, Frank, Holmes, Hall, & Witten, 2001), and has been an idea, if not by name, present in several pieces of work by the HCI community. One example of such a system is CueFlik (Fogarty, Tan, Kapoor, & Winder, 2008), a system that allows its users to rapidly create a model to assist them in ranking images based on their characteristics.

Unlike many of these prior works, however, IMT focuses on how the person engaged in the model building activity interacts with the learning system and how a teaching experience elicits explicit and implicit forms of human knowledge such as labels, features, rules, examples, etc. Generally speaking, teachers perform a range of teaching activities, including providing explanations and assessing student performance through observing student behaviors in various real or simulated situations. In the context of IMT, we say that a person acts as a teacher if they:

Plan (and update) the teaching curriculum. A teacher can decide how to solve a problem or how to teach. A teacher may choose what examples to teach with, choose when and how to evaluate the task against the task's goal, choose when and how to fix/update the model, decompose or merge concepts, features or tasks, etc.

Explain knowledge pertinent to the subject domain. One of the key aspects of IMT is that it advocates for teachers expressing rich semantic knowledge to a machine learner to make the teaching process efficient.

Review the learner's progress while integrating the given knowledge. As with traditional IML, teaching takes the form of a dialog, as the teacher's next action is influenced by the learner's current state.

When people do not behave as teachers

Our description of the role of a teacher, let us see how many roles people have in IML systems are not those of a teacher. Many activities require knowledge that does not relate to the subject domain for which one is building a model. These are not teaching activities. For example, a user

who is asked to directly manipulate algorithm parameters (such as number of learning epochs or learning rate) or think about ways to improve learning algorithms (such as changing a DNN's architecture) is not acting as a teacher because the knowledge transferred is not directly related to the subject domain but rather is specific ML knowledge. Some activities are implicit activities in which people have little or no agency with respect to the information collected by a learning system. For instance, an activity tracking systems that collect information from people without their knowledge and later uses information based on GPS traces to learn about the location of a person's "Home" and "Work". Some types of interactions, while related to building ML models, offer limited agency on the process outcome; thus we do not consider them teaching activities in the context of IMT. For instance, one-shot teaching scenarios in which a person feeds a massive number of labels to train a DNN has an extremely slow evaluation cycle and has further limitations on agency as it is typically not clear what to do if things go wrong. We note that active learning (Settles, 2012), defined as the (machine) student guiding the curriculum of examples to use for training, can provide support for a teaching activity. Active learning by itself, however, is not teaching as it gives curriculum agency to the algorithm instead of the teacher. Perhaps the simplest ML model-building process deserving of the interactive machine teaching moniker is an iterative process of (1) selecting and labeling of examples and (2) evaluating student learner performance using selected examples.

3.3. The iterative machine teaching process

IMT is an iterative process. In particular, it is one in which a person (or people) with a role of a teacher(s) interacts with a machine learner in order to transfer the knowledge of how to accomplish a task to a machine-learned model. In the previous section, we introduced a distinctive set of activities people in that role perform. In this section, we expand on these activities and how they connect to an IMT process, which we call the IMT loop. Note that we use the term "loop" to indicate that this is an iterative process and not to indicate that there is a rigid sequence of steps that people must follow. During this loop, teachers have opportunities to leverage their human capabilities and translate them into knowledge that they pass to the learner. Figure 1 illustrates how the teacher engages in *planning*, *explaining*, and *reviewing* activities as they interact with a learner, and with the goal of building an ML model. This figure highlights teacher activities for our discussion and is not intended to preclude iterative loops where machine interventions can support teacher activities.

Planning

The planning activity is one in which the teacher considers and identifies the materials that are useful for teaching. These can include examples, features, tests, etc. The nature of these materials depends on the goal of the teaching activity. For instance, if a teacher wishes to teach a learner to classify a web page into different categories, they might browse and search a large and diverse collection of web pages. Alternatively, if a teacher wishes to teach a robot to perform an activity such as navigating inside of a novel building, they might design a *training set* of alternative interior configurations by varying important factors such as layout, lighting, people, and decoration. What is important in the planning activity is for the teacher to identify a diverse list of examples or demonstrations to explain later and that are useful for expressing the task's goal while being challenging for the learner. In an iterative process, planning is also a moment of reflection where a teacher can

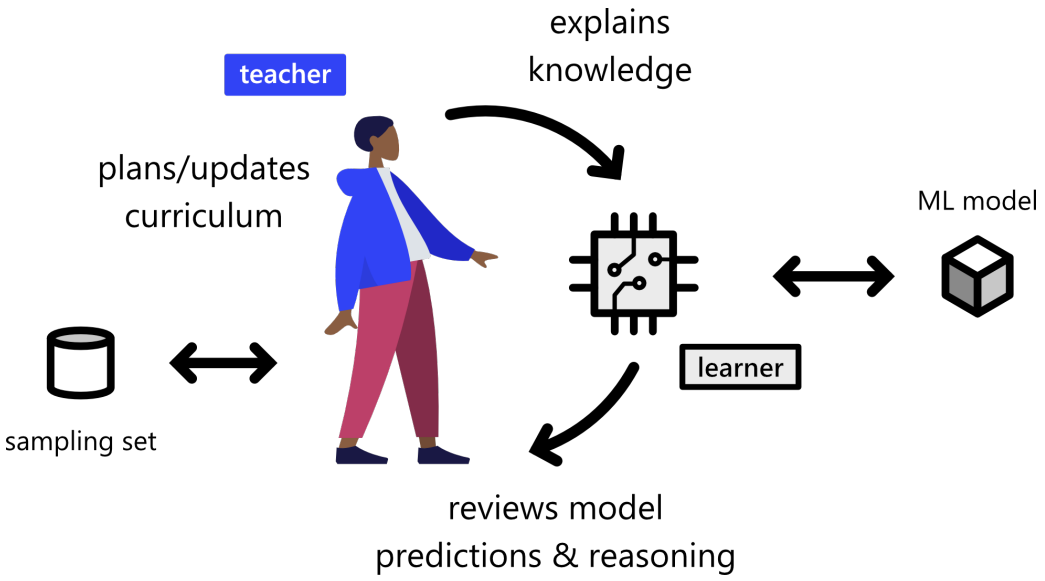

FIGURE 1. The IMT loop. During an interactive machine teaching session, there is a human teacher and a machine learner that communicate iteratively through a teaching language. While engaged in this process, teachers plan the curriculum, explain knowledge, and review the learner's state and model's predictions [3].

revise their strategy or revise prior decisions regarding concepts to teach. Here is where teachers assess if their concepts have evolved (Kulesza, Amershi, Caruana, Fisher, & Charles, 2014) and determine how their next steps should be adjusted accordingly.

Planning involves being aware of the types of data that the learner will encounter in the real world. We call this the *deployment distribution*. A critical component of the planning activity is the set of examples that a teacher has access for browsing, a set that captures the richness and diversity of the deployment distribution. If the teacher does not have access to a large, rich pool of examples, they must rely on their creativity and knowledge to construct suitable examples. In many scenarios, rather than accessing the full, real-world dataset, a teacher might have access to a sample of unlabeled examples. We call this subset of unlabeled examples drawn from the deployment distribution of the *sampling set*.

Ideally, the distributions of the sampling set and the data in the real world are similar (i.e., the sampling set is representative of the deployment population), but achieving distribution parity is difficult in practice. While the discrepancy in the distributions of sampling set and the deployment distribution is potentially problematic for traditional ML building workflows, it does not impede machine teaching. The most important requirement of the sampling set is that all types of examples expected to be present in the deployment distribution be represented. If representative examples are missing from the sampling set, the performance of the deployed model could deteriorate in unpredictable ways. As a rule, unlabeled data should be indiscriminately collected

[3]Humans illustration by Pablo Stanley, licensed under CC BY 4.0.

because the cost of storing data is negligible compared to the cost of teaching; we view selectively collecting only the data that is meant to be labeled as both risky and limiting. *A rich and diverse data set allows the teacher to browse it to express knowledge through selection*. It also allows the teacher to find examples that can be used to teach smaller, more specific sub-concepts (e.g., soccer) than the original concept that the learner already knows (e.g., sports). For instance, a teacher could decide to build classifiers for *bonsai gardening* (sub-concept) and *botanical gardening* (excluded concept) to be used as features to a gardening classifier. The sampling set needs to be rich enough to contain sufficient examples to teach such sub-concepts successfully. The sampling set can be updated or re-collected. Examples that have been labeled by teachers, however, should be kept forever because labels always retain some semantic value.

During planning, teachers concept decomposition (i.e., decomposing a concept into sub-concepts, or aggregating sub-concepts into a higher-level concept). Decomposition is a key form of knowledge people can provide and is useful not only to solve large problems, but also to share and reuse concepts among teachers. Taxonomies and hierarchies are examples of the typical outcome of a teacher exercising concept decomposition.

Explaining

The explaining activity is the activity by which the teacher explicitly provides the knowledge required to the learning algorithm to perform the task of interest. Not surprisingly, this activity is multifaceted. One facet involves describing what ought to be done in particular instances (e.g., how a web page should be classified, or reward an agent's behavior). This action is often called labeling. Also, the teacher can provide conceptual explanations of why particular decisions ought to be made. Conceptual explanations often involve other concepts (e.g., to understand an address, one needs to understand other concepts such as a city, state, and zip code). Thus, explanations often involve a decomposition of tasks or concepts. Other forms of teaching can involve the specification of rules, as in the case of underlying labeling functions (Ratner et al., 2019).

The explaining activity aligns with the points made by (Amershi et al., 2014), stating that people want to provide more information than just labels. Providing semantic knowledge (i.e., semantic features) to a learning system can lead to semantic models, where one can inspect the model's prediction process. Teachers can take advantage of these semantics, during the reviewing activity, to debug a model and find tractable fixes to prediction errors.

Critical to this activity is that the teacher has the tools to communicate the knowledge required to explain the desired behavior. This need translates to teachers having access to a *teaching language* that is sufficiently expressive, as well as a learning algorithm that can leverage the knowledge specified through this language and produce the desired ML model. It is through this language that a teacher provides descriptions of the behavior (e.g., labels) and the conceptual knowledge (e.g., features). Ideally, the teaching language should be simple and easy to learn and use by the teacher given the domain (e.g., text, signals, images). We assume that the required concepts can be expressed and learned through concept decomposition into primitives for different types of examples. Another important factor from both the human factors and machine learning perspective is the treatment of ambiguous scenarios. For example, a teaching language should give sensible ways for a teacher to express when they cannot decide what label(s) to assign to an example. Such language may take the form of an "unsure" value as a label, or the ability to skip or defer a decision, or instructing the learner to ignore an example altogether.

Reviewing

During the reviewing activity, the teacher evaluates the performance of the learned model and determines whether and how to continue to teach the system. This activity can include debugging erroneous predictions, assessing the quality of predictions, identifying problematic examples, identifying incorrect or incomplete labels that the teacher has provided to the learner, and building trust in the performance of the system in various use scenarios.

There are three types of prediction errors a teacher can find and choose to review during this activity. One of these types of prediction errors occurs when the teacher has not provided the learning system with the concepts to make appropriate decisions. We call this type of prediction error, a *concept blindness*. For instance, we want to build a web page classifier for identifying gardening web pages, but the current model classifies a rock garden as a garden contrary to the teacher's goal. In this situation, the teacher might want to teach the concept of a rock garden and to provide negative labels to examples of rock gardens indicating that the learner should not consider a rock garden as a garden. In addition to concept blindness errors, there are other errors in which the teacher has mislabeled the desired behavior for an example, or an error that can be corrected by adding a newly labeled document in the learner's training set (Meek, 2016).

In addition to debugging prediction errors, the goal of the reviewing activity is to enable the teacher to gain an understanding of the model's performance. Critical to this process is, again, the teaching language: the learning algorithm must communicate with the teacher by describing its algorithmic understanding of the example. Doing so enables a teacher to identify the nature of the errors and identify the best course of action to resolve them. This language needs to be shared between the teacher and the learner; the model communicates to the teacher using the same vocabulary, visual affordances, and encodings that the teacher, in turn, uses to express knowledge. We argue that a shared language is advantageous and leads to designs that are less complicated to be adopted and used by teachers.

4. LEVERAGING AND SUPPORTING HUMAN CAPABILITIES

Central to IMT is the principle of supporting tasks that are easy for most people to do, but currently beyond the reach of machines in the general case. Examples of these include summarizing documents, segmenting complex entities from images, and decomposing concepts into structures. For IMT, we look into supporting and leveraging people's ability to teach or transfer knowledge to another person and translate it to the task of teaching to a machine.

As described in the previous section, teaching occurs within a teaching loop (Figure 1). In this loop, teachers take actions based on their human capabilities. It is the role of a well-crafted experience to translate the knowledge embedded in these actions into information that the learning algorithm can leverage.

4.1. Through a constructivist lens

To identify what human capabilities to support, we look for clues from teaching theory. Strategies and approaches to teaching people are driven by theories about how people learn, so we use these learning theories to think about how to leverage and support people's capacity to teach. We engage in this process fully acknowledging that using human learning theory is not the most ideal

framing, as there are not yet machine learning theories that connect to models about how people teach. However, we see it as a useful way to ground IMT into existing cognitive theories. While several learning theories exist such as Behavioralism (Phillips & Soltis, 2015) and Cognitivism (Amsel, 1989), we look at Constructivism (Wadsworth, 1996) as a reference point, in part, due to the fact that others have already drawn connections between Constructivism and IML (Sarkar, 2016) (Nowak, Castellini, & Massironi, 2018).

Constructivism states that a student or learner builds new knowledge and models of the world based on (a) the knowledge they have, (b) their internal representation of the world, and (c) experiential input that challenges what they know. Constructivist theories are often attributed to Jean Piaget (Wadsworth, 1996), and while not without criticism, they are recognized as significant theories influencing pedagogy and computer science education (Ben-Ari, 1998). This constructivist view aligns with the teacher's mental model for the computational learner in IMT: the teacher incrementally increases the student's concept space by introducing new knowledge that builds upon existing ones and presents the student with examples that challenge its current model of the world, leading to teaching actions that update the model.

4.2. Human capabilities and teaching

Through this constructivist lens, several actions or activities happen during a teaching session. We inspect these activities to reveal the underlying human capabilities that seem fundamental for teaching.

Knowledge grows through the perturbation or challenging of the student's current concept structures. The teacher is a facilitator to this process. This process relates to the IMT activity of planning and teaching, where a teacher selects meaningful examples to label, which they expect will challenge the model's predictive power. During this activity, a teacher recalls examples they foresee will challenge the model (i.e., employ *foresight) and assign appropriate label values (i.e., apply* judgment).

There is a mutual, continuous, and interactive assessment between teacher and student. This activity relates to the interactive nature of the IMT loop, and in particular, to the IMT activity of reviewing. While current learning algorithms do not truly assess a teacher, machine-in-the-loop interventions such as active learning demand a teacher's *judgment*. Likewise, a teacher's assessment of a model includes inspecting and correcting prediction errors though re-evaluating the appropriate label values (i.e., employ *judgement*) and discover previously unrevealed knowledge (i.e., gain *insight* from errors), respectively.

The teacher must be able to perceive how the learner's intelligence evolves. This activity also relates to the IMT activity of reviewing. Here a teacher evaluates how a model performs on seen and unseen examples. During this activity, a teacher applies *judgement* to establish when a model's predicted label is incorrect and *insight* to understand why. During this assessment, the teacher's understanding of the problem or target concept can also evolve (Kulesza et al., 2014; Felix, Dasgupta, & Bertini, 2018), leading to the process of developing, collapsing, expanding, and organizing their concept structures (i.e., employ *sensemaking* (Pirolli & Card, 2005)) that they possess and have passed to the learner.

The learner builds knowledge from knowledge building blocks or examples. This process relates to the IMT activity of teaching, where the learner receives knowledge that can take the form of labels, features, examples, rules, etc. If the source of this knowledge is teachers, they must apply

judgment to assign it to a relevant class or hierarchy.

Four main human capabilities come forward from these activities. These are *judgment*, *insight*, *foresight*, and *sensemaking*. We now elaborate on these capabilities in the context of two types of IMT scenarios: *classification* and (entity) *extraction* for text documents, important classes of problems and data. We define classification as the task of assigning one or more labels to a whole document. We define extraction (sometimes called segmentation) as the task of assigning one or more labels to a *part of a document*. While these four capabilities are also applicable to other teaching scenarios outside of supervised ML, such as RL, we use these scenarios to provide concrete examples of these capabilities. We will also use these scenarios in the remainder of the document, to anchor concepts and ideas leading to Section 5.

Judgment

Judgment is a capability that enables people to make decisions about the label value (e.g., true, false) of a particular example or document. It relates to the general and specific subject-domain knowledge a person has, which helps them to recognize a particular concept, entity, or pattern. For example, when a person teaches a binary classifier that determines if a text document is of SPORTS or not, judgment allows a person to assign a value of *true* or *false* to the label SPORTS. When a person teaches an extractor that can retrieve addresses from a text document, judgment allows the person to label parts of a text document containing an address as *true* for ADDRESS and the rest of the text document as *false* for ADDRESS. This tuple of $(document, label, value)$ is a fundamental input to any supervised learning classification and extraction algorithms. Judgment also lets people make decisions about what the state of a system is in, which can suggest what actions need taking. For example, a teacher is training a classifier for US addresses that currently has a training accuracy of 85%, while in prior occasions the teacher decided that this was too low an accuracy and made them fix prediction errors on the training set; the teacher now deems that they want to finish labeling new examples that include PO boxes, as none of the prior training data included them.

Insight

Insight is a capability that enables people to analyze and synthesize knowledge relevant to what they see. This ability relates to recognition and leverages a person's subject-matter knowledge. Insight helps people to parse information at different levels of granularity. **Insight directly translates to teachers articulating or refining semantic features** to the learning algorithm, often at times when concept blindness errors need to be fixed. Besides document-label values, features are another fundamental input to any learning algorithm. In the context of machine teaching, we use the term *semantic feature* (Jandot, Simard, Chickering, Grangier, & Suh, 2016) to describe a function that, given a document, returns a data structure that a learning algorithm can use and that represents a human concept. Examples of semantic features include the length of a document, the number of times words in a list appear in a document, the presence of two or more faces in an image, etc. In the case of a SPORTS binary classifier for text documents, for example, insight allows a teacher to correct a false positive error of a model erroneously giving a high prediction score to a document containing the word "cricket" in the context of an article about insects. By inspecting the document, the teacher realizes that the word "cricket" is not exclusive to the concept

of SPORTS, and creates a semantic feature to capture a notion of INSECTS. Later in the article, we provide specifics about how a teacher can do this.

Foresight

Foresight is a capability that enables people to articulate knowledge solely from prior subject-domain knowledge without the need to see a particular piece of information. This ability relates to recall and subject matter expertise, which can help a person articulate knowledge about a topic. **Foresight can indirectly translate into labels, features, and other forms of inputs** to a learning algorithm. During the teaching loop, a teacher uses foresight to select a good document to label that the model has not seen previously. For example, a teacher building a FOOD RECIPE classifier, knows that the word "recipe" is a good signal to look for in a semantic feature. However, they also know that it can appear in many non-food contexts such as "recipe for disaster" or "numerical recipe". Because of this ambiguity in the word "recipe", they make sure to find example documents for a variety of contexts where "recipe" appears and label them appropriately. Teachers can also use foresight to bootstrap a learning algorithm with semantic features before seeing or labeling any documents. For example, a teacher building a COOKING INGREDIENT extractor knows that it will be beneficial to create a semantic feature for the concept VEGETABLES, which they articulate as a function that returns the number of occurrence of vegetable terms (from a set of words they import or curate) contained within a document. Foresight can also be used to help a teacher propose an initial decomposition or hierarchy of a predictive model's output (which we call model *schema*). Following this example, a teacher might decide, before seeing any documents, that a COOKING INGREDIENT is an entity composed of a number, a measuring unit, an ingredient name, and an optional preparation step (e.g., 1 Tbsp butter at room temperature). Foresight is a capability that evolves, and it is influenced by what a person sees and experiences. The knowledge that foresight produces can overlap with the knowledge that insight or sensemaking produces. We, however, explicitly specify foresight as the knowledge that emerges directly from a teacher's mind without being directly triggered by seeing a particular detail in a document.

Sensemaking

We use the term sensemaking in the context of the work by (Pirolli & Card, 2005). Sensemaking is a capability that enables teachers to interpret the information they have seen so far and to translate it into a hypothesis regarding ways in which information and concepts relate through higher-level concepts, structures, or relationships. Sensemaking and insight are closely related as both tap into a person's general knowledge and subject-matter expertise. We view sensemaking as different from insight. Insight supports sensemaking, and sensemaking is a higher-level human capability that leads to forming theories about how concepts and information across multiple documents relate and can be organized. Foresight also supports sensemaking. However, while foresight is independent of particular documents, sensemaking relies on the traceable evidence within a set of documents. **Sensemaking often produces knowledge that takes the form of structures, relational patterns, or the decomposition of a concept into sub-concepts**. This knowledge can, in turn, produce information that helps a learning algorithm to produce an ML model in efficient ways. For example, after observing several documents, a teacher may realize that sentences can convey the result of a sports match: the names of two teams are separated by a certain number of words (e.g., *"**Real Madrid** had a great game, where they defeated **Barcelona** 4-2"*. After seeing

more documents, the teacher may refine that structure into one where two sports team names are separated by a certain number of words with a sports-related verb, such as *"defeated"*, between the two team names. In another example related to the previous section, a teacher might need first to see several recipe candidate documents to realize that they need to articulate the concept INGREDIENT that expresses a structure including number, measuring unit, ingredient name, and an optional preparation step. A teacher can do this by creating an extraction sub-model (with the above output schema) that they can then import as a feature on the recipe classifier.

Sensemaking is crucial for concept decomposition activities. One might argue that breaking a complex concept into simpler ones is at the core of well-developed problem-solving skills. For example, after observing several documents, a teacher realizes that a **Recipe** document is usually made up of sections such as **Instructions** and **Preparations**, as well as often containing information about **Nutrition** and **Servings**. The teacher then decides that articulating those concepts and the relationship among them (e.g., a recipe is made out other parts, and some of those parts are mutually exclusive) is useful knowledge for the learner to know.

Existing work shows that tools can support sensemaking. For instance, the work by (Kulesza et al., 2014) provides a mechanism to support *concept evolution* through structured labeling tools. In these tools, model builders have access to a visible structure they can edit, and that helps them make sense of the concepts at play.

4.3. Supporting human capabilities

The above capabilities are as effective as the resources and tools that the teachers have available with which to apply these capabilities. As we look into practical implementations of IMT systems, we consider a number of human tasks and needs relevant to supporting these capabilities. In the case of judgment, besides providing labeling functionality, teachers need to defer decisions while their concepts form or evolve. When applying judgment about the state of the system, teachers need meaningful and actionable feedback that lets them decide what to do next, such as testing, correcting errors, find and provide more examples, etc. To support insight, it is important that teaching experiences present documents and information faithfully and, when possible, in the context of other documents. When used to resolve prediction errors, insight benefits from teachers having awareness about what the ML model sees in a document (i.e., what information from a document a model uses to make a prediction). To support foresight, it is important that teaching experiences give teachers a rich language that lets them describe concepts and retrieve the documents or information foresee in their heads. In some circumstances, teachers should be able to create, examples or testing sets based on foresight, so one should consider making this functionality available. Finally, one supports sensemaking by giving teachers the language and mechanisms to express and update concepts, structures, and relationships.

5. INSIGHTS FROM AN INTEGRATED TEACHING ENVIRONMENT

Choosing interactive methods and artifacts as instruments is particularly important when studying IML solutions. Sometimes these instruments take the form of design probes that focus on a particular aspect of a larger problem (Hohman, Head, Caruana, DeLine, & Drucker, 2019). In

other cases, one can choose to simulate a system or behaviour through a wizard-of-Oz protocol (Maulsby, Greenberg, & Mander, 1993). In this section, we elaborate on how we used a design probe and a wizard-of-Oz protocol in two studies to observe and understand different aspects on how subject-matter experts with no ML expertise can use IMT to create ML models (extractors and classifiers) for text-documents[4]. To support these tasks and future research on IMT, we implemented an *integrated teaching environment* (ITE) called PICL. In this section, we describe PICL and how it supported our design probe. We later introduce a variation of PICL we call MATE that supports our wizard-of-Oz protocol.

5.1. PICL: a platform for interactive concept learning

We designed PICL's to as a platform to help our IMT research, thus we aimed at it providing teachers with an interface for building ML models by supporting their capabilities while engaged in an IMT loop. Teachers bootstrap a teaching session in PICL by creating a teaching project where they import their text data source and specify the type of ML model they want to build. At any point during a teaching session, teachers can export or publish the latest in-progress ML model for others to use as a runtime to be imported in their software solution. As its functionality is common to many IDEs, we do not go into details about the ITE's UI for project and session management, including setting up data sources and exporting models. Once teachers define the sampling set and type of the model they want to build, the system presents a teaching interface (Figure 3) that allows them to perform the following essential teaching tasks:

Defining an Entity Schema. Essential to teaching an extractor is the presence of an *entity schema*, a hierarchy that describes the main entity to extract and its sub-entities, if any. At any point, users can edit this hierarchy, enabling them to modify the target specifications of the extractor model. Editing the hierarchy is often necessary as users become more familiar with the set of documents they encounter while teaching and a way of supporting concept evolution (Kulesza et al., 2014). For example, through foresight, a user might define an ADDRESS as being made out of a STREET, CITY, STATE, and ZIP CODE only to later realize, through sensemaking, that sometimes an address can have a P.O. BOX. Teachers rely on sensemaking and foresight during this task.

Sampling Documents. Picking the right document to label has a positive effect on the training of an ML model. Ideally, one wants to choose a document that, when labeled, provides useful information to the underlying learning algorithm. For example, a user might want to first label addresses they know, such as ones that contain ZIP CODE 10020. Teachers rely on insight and foresight during this task. Figure 2 describes alternative samplers that are available to the teacher in PICL. Samplers are most effective over a large sampling set that captures the distribution of the data that a model will find once it is deployed. In turn, small, unrepresentative datasets produce models that might capture this training set well, but then fail in real-world scenarios. The study of different exploration strategies in support of a teaching flow is a substantive topic of discussion, which is beyond the scope of this article. (Settles, 2012) provides relevant information about active learning and other approaches for choosing training examples.

[4]While IMT concepts can apply to other types of documents such as images, we choose to initially focus on text as it remains an important class of documents ML models applies to.

[10]In addition to the in-progress model, PICL can also train other models, such as bag-of-words, against the same training set behind the scenes. These models can be used to sample for documents where the models' predictions disagree.

[10]In addition to the in-progress model, PICL can also train other models, such as bag-of-words, against the same training

Sampler	Description
Keyword Search	Get a document containing one or more keywords.
Random	Get a random document.
Uncertainty	Use an active learning strategy to get a document near the decision boundary.
Explanation-Based	Use an existing concept to get a document containing it.
Possible Errors	Get a document the model is likely predicting incorrectly, when contrasted with an alternative model [10].
Predicted Positives	Get a document the model predicts as positive.

FIGURE 2. Sampling Strategies supported by PICL.

Labeling Documents or Entities. Labeling describes the process of, given a document, annotating it as a whole, or its parts with the value of a set of labels. For the example above, this task can include marking consecutive words as ADDRESS and not ADDRESS given a text document. During this task, teachers rely on judgment, which allows them to tell if what they are seeing corresponds to an entity or not.

Providing and Editing Features. There are cases where the model predicts an entity incorrectly because of a *concept blindness* (i.e., the model does not know what to look for to decide if something belongs to a class). To address this issue, a teacher needs to provide features to the machine learner. Using the example above, a teacher might provide a concept for ZIP CODE that detects if a token consists of five digits. Teachers rely on sensemaking, insight, and foresight during this task.

Reviewing Predictions. At any point, a user can examine training or prediction errors. A user needs to detect these errors and decide what to do next. This debugging task and the resulting fixes often involves producing additional labels or features (Amershi et al., 2015; Meek, 2016). Teachers rely on insight and foresight during this task.

Related Systems

PICL stands unique among other IML systems. We designed and implemented it as a comprehensive teaching environment that follows IMT principles: It provides an end-user text classification and extraction teaching experience, where no coding or ML experience is required. It supports the communication of semantic knowledge beyond labels in an iterative way, which in turn allows teachers to refine or evolve concepts. PICL produces semantic models allowing teachers to inspect predictions in order to debug and fix the errors. It provides immediate feedback about how teaching actions affect the model's performance. It uses machine-in-the-loop interventions to amplify a teacher's labeling and sampling capabilities.

Systems that aim at achieving the same goal of helping end-users to build text classification or extraction ML models compare only partially. Most cloud services such as IBM's Watson Studio (IBM, 2020), Amazon Sagemaker (Amazon, 2020), or Google's AI Platform (Google, 2020a) take a one-shot training approach, where an end-user feeds a system a large number of labeled examples before a model is trained. These solutions do not permit for concept evolution to oc-

set behind the scenes. These models can be used to sample for documents where the models' predictions disagree.

cur organically. Debugging experiences in these systems are also uneven, and analyzing how to fix errors can involve knowledge outside of the subject-matter essential for the task's completion. Prodigy (Prodigy, 2020) describes itself as supporting "Machine Teaching" experiences and is aimed at end-users with knowledge of some ML as well as coding in Python. Azure ML Studio (Azure, 2020), also provide end-user ML services that necessitate coding knowledge in order to take advantage of debugging features using Jupyter Notebooks (Jupyter, 2020).

LUIS (Microsoft, 2020) is the result of early ideas on IMT. Its scope and capabilities, however, are different from PICL's. LUIS focuses on short utterances, and, while interactive, it does not have advanced sampling, featuring or debugging capabilities.

Related research systems can share some of the same ideas, but are not comprehensive as they focus on how the variation of one or more variables affects the result of a study. Kulesza er al.'s system (Kulesza, Burnett, Wong, & Stumpf, 2015) for labeling emails shares similarities with PICL's teaching language regarding concept explanations, but the type and amount of feedback they allow end-users to provide is ultimately not the same. The Abstrackr system (Wallace, Small, Brodley, Lau, & Trikalinos, 2012) for classifying relevant abstracts has similar goals to PICL's, as well as an interactive training workflow. However, their system provides little to no agency in debugging or ways to learn from user-provided knowledge beyond labels. The Exploratory Labeling Assistant (Felix et al., 2018) uses machine-in-the-loop interventions to assist in the task of discovering, curating, and evolving labels for a collection of text documents. While related, this system is not a teaching environment or deals with the goal of building a machine-learned extractor.

PICL's General Teaching Interface

PICL's teaching interface is divided into three areas, each dedicated to specific teaching tasks. Figure 3 illustrates these three areas (left, center, right) of the main interface.

The left section supports teachers reviewing predictions by revealing the current model's behavior against the set of labeled documents (training set). In this region of the UI (Figure 3-A) teachers examine prediction errors using a *model tracker* interactive control (Amershi et al., 2015). In our implementation, model predictions are visualized as unit boxes corresponding to labeled documents (if the model is a classifier) or a segment within the document (if the model is an extractor). Teachers inspect a labeled document (for a classifier) or part of a document (for an extractor) by selecting a box in the visualization. Doing so opens the corresponding document in the middle area of the UI. We describe this control in more detail in later sections.

The UI's center region is where teachers have access to document sampling controls (Figures 3-B), as well as the ability to (re)view and label a document or segments within a document. PICL provides teachers with a selector to pick among different sampling or exploration strategies. These strategies, described in Figure 2, are a representative set of the different ways one can find useful examples to label. Below the sampling controls is the document view area, where teachers can inspect a document, and depending on the type of model being built, label a whole document or parts of it. In addition to applying judgment to decide if a label's value is positive or negative, PICL allows a teacher to defer judgment and be *undecided* about how to label a document. Deferred judgment or undecided documents appear as blue boxes in the model tracker. The ability to decide later helps concepts evolve (Kulesza et al., 2014; Felix et al., 2018) as the teacher gains insight from future document inspection and labeling actions. The document viewer, or labeling control, also serves as an interactive canvas that the current model-in-progress can enhance though decorations

based on what the model is currently predicting, and what concepts it thinks are relevant. We will describe in more detail this active, mixed-initiative control later in the article.

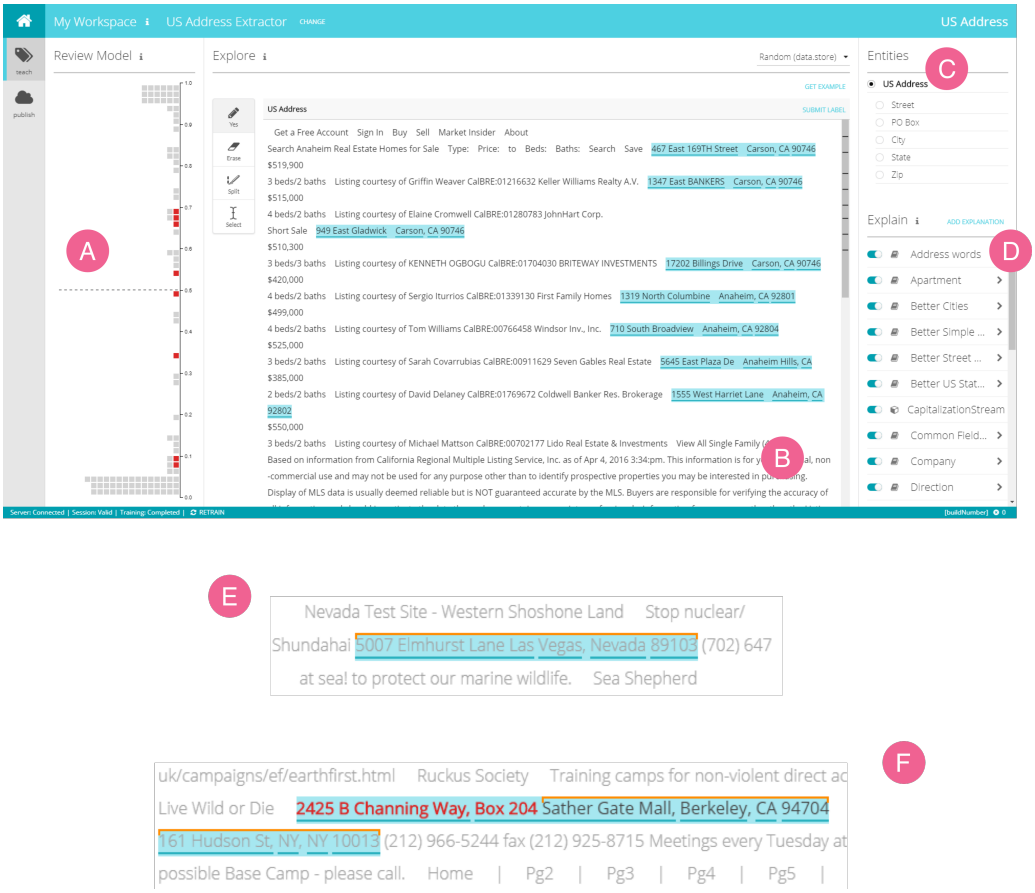

FIGURE 3. PICL's main interface. A - The *model tracker*. B - The *document labeling control*. C - The *entity schema* editor. D - The *explanations panel*. While in *reviewing mode* the labeling control also displays predictions vs user-submitted labels. E - Example of a teacher's label and a system's prediction (as an upper bracket) in agreement. F - Example of a disagreement between a teacher's label and the system's current prediction: the upper bracket does not match its corresponding label, also we emphasize in a red contrast color the words where there is disagreement.

The UI's right area is where teachers can articulate semantic concepts (used as features for the learning algorithm). In PICL, we do not directly present users with the notion of an ML feature. Instead, we use the notion of *concept explanations*. While the functionally is the same, we believe that presenting the features in this way eases understanding and use by non-ML experts. PICL provides a control (Figure 3-D) to let users add explanations while teaching. PICL supports three types of explanations: *basic explanations*, *patterns*, and *related concepts*.

Basic explanations are simple lists of n-grams. These basic explanations translate to a function that activates (returns a non-zero value) if any of the n-grams in the list matches a string in the document. For example, when building an address extractor, a user might choose to explain the concept of "US State" by specifying a list of US states. Users create or edit these basic explanations by typing or pasting a list of n-grams into a text editing area. When specifying basic explanations, teachers can indicate to the system to consider stem words to match plural variations, as well as digit identity patterns so that the n-gram "000" match any three-digit string. Patterns describe a structure among a set of existing explanations. In particular, PICL only implements simplified regular expressions that capture the proximity of different concepts. These expressions can describe part of documents like the region defined by two sports team names that are separated by no more than a certain number of words, one of which is a sports-related verb. Related concepts are existing models built in a separate session and later referenced by the teacher. For example, a teacher might build an ingredients extractor to be used as an explanation for the INGREDIENT concept in a cooking recipe extractor. Related concepts are a way for teachers to decompose a larger concept into smaller ones.

The right region of the interface is also where teachers building extractors define and can edit a model's main output schema (Figure 3-C). PICL provides a simple schema editor that lets users add, remove, and rename entities in a hierarchy. This control is also where a teacher selects the entity they are currently focusing on, which provides context to the teaching interface (i.e., the selected entity is the one the user will be labeling for, and the context for the model tracker (on the far left)).

Designing mixed-initiative teaching interactions

One of the characteristics of IMT is that a teacher and a learner are engaged in an exchange of information where knowledge and information can flow in either direction. An important aim of IMT is that of improving the productivity and efficiency of teachers building an ML model. We aim to do that by amplifying the teacher's labeling and debugging capabilities, through *mixed-initiative* interactive controls that use the current state of the model being trained to provide helpful predictions. We use the term mixed-initiative to denote experiences shaped by the interplay between a user's direct interventions/manipulations and those of an (interface) agent. (Horvitz, 1999) provides a seminal list of principles to guide the design of these types of experiences. We now describe these PICL controls in more detail.

Document Viewing and Labeling Control

The document labeling control, at the center of the teaching UI (Figure 3-B), is where teachers can view a document and can specify as a whole what class it belongs to (in the case of a classification problem), or what parts of it are examples (or not) of a particular class (in the case of an extraction problem). In the cases a teacher is building a (binary) classifier, the control lets the teacher read the document and judge if the document belongs (thumbs up) or not (thumbs down) to the problem's class. In turn, the system presents information about the current document in view in two ways. First, the system can highlight parts of the document where an explanation activates and lets teachers inspect what explanation corresponds to a particular highlight as they hover on it. In this way, the document labeling control is letting teachers know what the model-in-progress can and cannot see. Second, the system displays the current model's prediction score for

the document in view, as well as pre-selecting a labeling action, based on this score being above or below a decision score of 0.5. As the model improves, one expects that these pre-labels will be more accurate, leading to teachers only having to submit a label, instead of selecting and then submitting one.

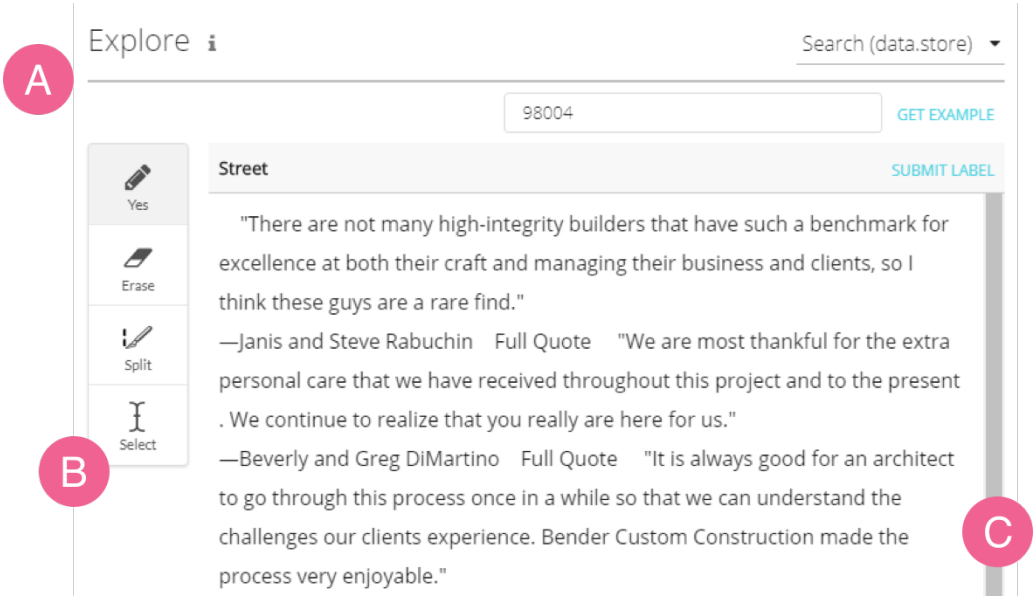

FIGURE 4. Sampling and document labeling controls. A - sampling strategy selector, in this case *search*. B - labeling toolbar. C - document labeling area.

In the cases a teacher is building an (entity) extractor, the labeling control and actions take a different form and pattern. For extractions, we designed this control following a *painting tool* metaphor similar to systems such as Adobe Photoshop, where a user chooses a tool from a palette and applies it to a part of a working area. The labeling control consists of two main parts: the labeling toolbar (Figure 4-B) and the document area (Figure 4-C).

The toolbar lets users pick tools to perform different labeling operations; label a part of a document as a positive example (pen tool) or as a negative example (eraser tool) of an entity, and to split a labeled part of a document into two (cutter tool). It also provides a *select* tool for text selection in a document to accommodate future contextual operations such as copying text.

Labeling is always relative to an entity. The teacher chooses the entity in focus with the entity schema component (Figure 3-C). For example, while the entity ADDRESS is selected, the user applies the pen tool to mark a consecutive series of words in the document. Once marked, these words are highlighted to indicate that they are a positive example (label) of ADDRESS (Figure 6). The words without a highlight are negative examples of the entity in focus. If the teacher switches focus to another entity, say STREET, the previously highlighted part of a document becomes underlined (Figure 6) to show that it has been positively labeled but not as STREET. Users can inspect the current state of labels over words by switching the entity in focus to look for the highlighted

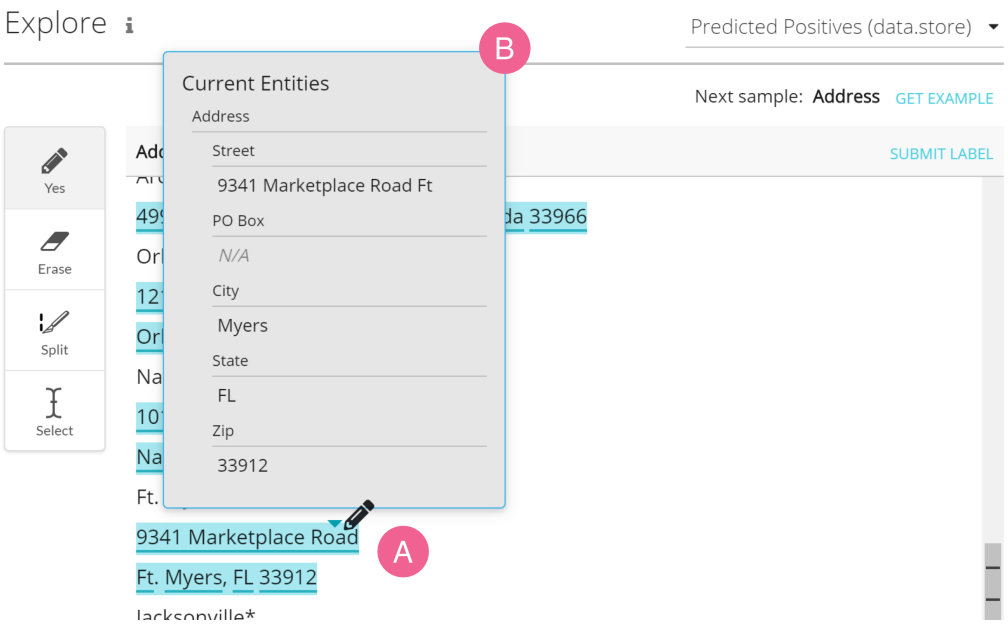

FIGURE 5. Pre-Label Inspection. This figure illustrates how a teacher can hover on a pre-label (A) and inspect the model's prediction (B).

words or by hovering the mouse over the word to reveals a pop-up area that lists how words fit in the global schema (Figure 5).

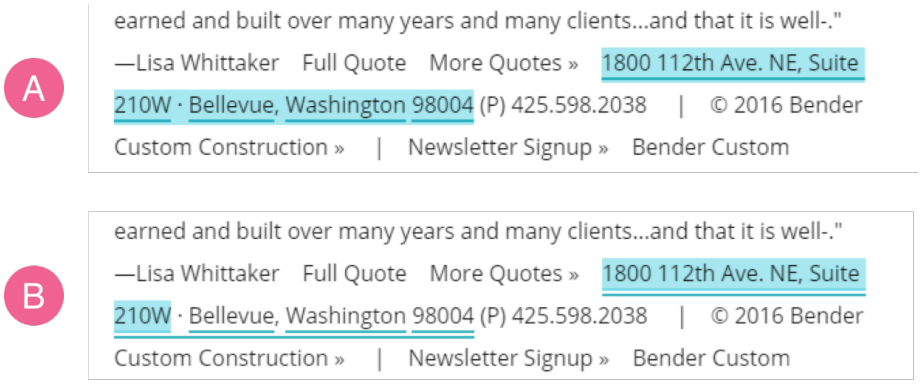

FIGURE 6. Labeled and pre-labeled entities. A - The system displays positive labels corresponding to the currently selected entity (*address*) as highlights. Positive labels for entities not in focus appear as underlines. B - (Focus on *street*) The system shows positive labels of *street* as highlights, and positive labels of other entities as underlines.

When users are satisfied with the labels on a document, they submit them to the system to be used as training data. This submitted document consists of the label values about all entities for all words in the document: all the positive labels (displayed as markings on the text) as well as the negative ones (displayed as text without markings). Once a user submits a document and its labels, the system uses the currently selected sampling strategy to bring a new document that takes the place of the previous one.

The labeling control uses the available ML model to assist in two ways. First, the control displays the model's highest probable predictions as pre-labels. These pre-labels appear as highlights and underlines as if another teacher has marked them on the document. This is not a mode that suddenly becomes active, in all previous documents with no pre-labels visible, the system was just predicting everything as *not an entity*, for every entity. Depending on how good these predictions are, the user can choose to correct them using the pen (to explicitly mark missed positive labels), eraser (to remove false positives), or cutter (to split an entity that is actually two) tools. Second, the system reacts to the user's labeling and corrections by updating[5] the most probable pre-labels consistent with what the model would predict given these user's new labels. As an example, suppose there is an address within a document that is not predicted as ADDRESS by the current model. The user then labels the first word in the address as ADDRESS, an action that the system follows by providing the most probable labels that are consistent with that partial constraint. We see this ability of the system to provide new predictions without the need to retrain the model as a powerful enabler, which can translate to the system rewarding small user interactions with a full 'auto-completed' labeled entity. It follows that this control's behavior can significantly accelerate the user's labeling throughput.

Using Pre-Labels to Test Performance

One of the core tasks when building an ML model is to have a sense of how well it will perform in the wild. A common practice is to measure this by using some form of labeled test set, but in many scenarios, these are not available, or they might not reflect the distribution of the data in the wild. Because of the above, PICL provides users with alternative methods to assess a model's performance.

One way is to use a *uniform* random sampling strategy over a predefined test set; the observed pre-labeling errors will be representative of the model's performance. In other cases, the entity to extract is rare, and sampling at random might not capture it, making it difficult to measure performance. To address this, we provide a sampling strategy that samples uniformly over the set of predicted-positive instances of the extracted entities. As users see the accuracy of these samples, they get an estimate of the precision of the model. Furthermore, if one multiplies this precision by the number of predicted-positive instances, a quantity we call *reach*, that measures how many examples in the sampling set will be correctly labeled positive. This quantity is proportional to the model's recall. For example, the teacher can sample uniformly from documents that are predicted to contain a *Street* label and then count the number of predictions that are correct.

Model Tracker and Debugging

As teachers label more documents, the system updates its model tracker (Figure 3-A). In this

[5]The specific details of how this is accomplished are beyond the scope of this article. Retraining the model can be expensive, but it is possible to use the model conditional on the updated information using a Viterbi-like algorithm.

control, we extend (Amershi et al., 2015) to the task of information extraction and modify parts of its design to help users focus on unit boxes of interest (i.e., disagreements between teacher labels and the model predictions). To our knowledge, ours is the first application of this type of control in a working interactive ML system, which we later study and observe in use during an entity extraction teaching task.

The model tracker displays a histogram-like structure of *unit boxes* in horizontal bins of fixed height. Each box represents a labeled document (in the case of a classifier) or a part of a document (in the case of an extractor). In PICL, the referred labels correspond to the entity currently in focus.

The position of the boxes is based on a confidence score for the part of a document it corresponds to, which can take values between zero (bottom row) and one (top row). Our confidence score uses the model's prediction score for the part of a document as a parameter; it is a modification to the confidence scores of Culotta (Culotta & McCallum, 2004) to support handling variable-length parts of a document[6]. The display area also shows a dotted horizontal line at the decision value (currently held at 0.5) that separates positive from negative predictions. Boxes with similar scores stack horizontally in the same bin. The control renders unit boxes in a solid neutral color (gray) when the prediction score corresponds with the teacher label value (i.e., for neutral boxes, the decision line separates true positive labels (above) and negative labels (below)). If the prediction score disagrees with the teacher's label, the system displays the unit box in an accent color (red) easily distinguished from the neutral one used for agreements. Contrast boxes above the decision line represent parts of the document the system believes are positive, and the teacher labeled as negative; boxes below the line represent parts of the document the teacher labeled as positive, but the system predicts as negative. This design makes these boxes stand out and addresses accessibility concerns regarding color separation while expressing full prediction semantics for a given entity (i.e., true positives, true negatives, false positives, and false negatives). This control serves the purpose of helping teachers not only have a visual snapshot of the current state of the model but also provides them with a hit list of documents to review.

Reviewing Documents and Labels

When users select any of the unit boxes on the Model Tracker, the system enters a *Review Mode*. In this mode, the labeling control displays the corresponding document, which the system decorates to reveal the user's labels against the current model's predictions (Figure 3-E,F). In this way, users can review and fix any prediction errors they see in the document. In addition to the above, highlights in the document let teachers see what the learning algorithm sees in terms of concept explanations, thus fostering insight as to the rationale of a prediction.

In the simple case where a teacher is building a classifier, the document labeling control shows the model's prediction through its numerical score.

During an extractor building session, the document labeling control uses the same visual style for displaying a teacher's labels (Figure 3) and displays predictions for the entity in focus as an upper bracket designed to help users quickly see when it either matches or not, its corresponding label given by a teacher (Figure 3). While correcting labels within a document, PICL can display new predictions and pre-labels based on the current model plus the newly added corrections. This type of system's behavior is one of the main ways in which mixed-initiative can manifest to help teachers being more efficient. By design, PICL emphasizes the words corresponding to a disagreement unit box in its accent color, if such a box is selected (Figure 3-F). If users identify

[6]The technical details of this implementation are outside the scope of this article.

prediction errors that likely stem from a concept blindness in the model or from a lack of labels, they can add a new or edit an existing concept explanation or exit reviewing and choose to label more documents.

Concept decomposition support

PICL supports concept decomposition through several of its controls. We differentiate between two types of decomposition: *output* and *input* decomposition.

Output decompositions occur when a teacher applies sensemaking to the output space of an ML model (i.e., the entity schema that an ML model detects and produces as output). Building on the previous example, a teacher decides (at once, or incrementally as they teach) that when building a U.S. address extractor, the model should extract and output an ADDRESS entity, made up of the sub-entities PO BOX, STREET, CITY, STATE, and ZIP.

Input concept decompositions occur when a teacher applies sensemaking to the input space of a learning algorithm in order to explain to the system what to look for when predicting the presence of an entity. This action might require introducing new concepts about parts of a page such as TITLE or FOOTER as sections where an ADDRESS is present or not. Teachers leverage the explanations area of the PICL interface (3-D) to decompose concepts in this way. Another way to decompose the input to the learning algorithm is to decide how to integrate or separate the appropriate data sources leading to a training set. For example, when building a recipe classifier for web-pages, the teacher can decide to use as input the HTML markup document, the rendered final webpage, or both.

Hiding Learning Parameters

Following the principles outlined in (Simard et al., 2017), PICL explicitly hides the particulars of its underlying learning algorithms. In the IMT perspective, teachers have agency on the learning process through the articulation of subject-matter knowledge, not the selection of a learning algorithm or its parameters. The only meaningful effect a learning algorithm has on the experience is the time it takes to train, as it can impact the flow of teaching. To build extraction models, PICL relies on a standard conditional random field (CRF) with a hierarchical state machine that ensures the predicted labels obey the constraints defined by the main entity schema. PICL also relies on standard logistic regression to build classification models. The implementation of these learning methods took a few seconds to retrain on an x86 PC using an Intel i7 processor running at 2.6GHz with a total of 16Gb of RAM.

UI Limitations

The current implementation of PICL does not preclude teachers from defining large numbers (hundreds) of features and entities, as well as sample and train over a large number (tens of thousands) of documents. The current interface, however, can make handling large sets of features and document sets challenging. For example, scrolling linear lists or wide & deep hierarchies could become cumbersome to interact. Similarly, the model tracker's usefulness, in terms of visual complexity

[6]We will discuss in Section 6.4 how the MATE system (Wall, Ghorashi, & Ramos, 2019), a variation of PICL, was used to study classification scenarios.

and interactivity, might become hindered once thousands of documents and labels are at play. While currently, PICL supports foresight through its sampling controls, future designs could allow for a richer document querying language that allows finding particular documents in a very large set. There is a case to be made about an ITE that leverages knowledge decomposition to limit the number of concepts that teachers have to manage at a time to a certain manageable size, thus forcing decomposition. Similarly, one can think in deconstructing the model tracker into separate interactive visualizations that scale better with larger document sets. For example, to guide teachers towards what errors to fix, a UI can presents a list of labeling disagreements between teacher and learner or predictions with uncertain scores, which can be smaller and more manageable than dealing with the overall training set. These issues are beyond the scope of our initial explorations and are part of the longer IMT research road map.

5.2. A design probe of interactive machine teaching

We used PICL as a design probe to see how well our design decisions in PICL helped non-ML experts during the process of building an entity extractor. We observed the labeling and reviewing activities of the teaching tasks, as we were interested in observing how our mixed-initiative controls affect the teaching experience. In particular, we focused on collecting observations and gaining insight to address the following research questions:

- RQ1.1 - What are unforeseen usability and design issues for IMT activities?
- RQ1.2 - How do machine-in-the-loop interventions support and affect IMT?

Participants

We required participants to not have built ML models in their professional capacity. This did not exclude people that used models built by others. We also required participants to be familiar with the format of U.S. addresses. Our participant pool consisted of 8 males and 2 females, with ages ranging from 25 to 54 years old. All were employees at a major technology company, and eight had an advanced degree. Regarding their familiarity with information extraction, all subjects reported either "I didn't know very much about it" or "I was familiar with the basic concepts". Regarding their experience with ML, two participants reported "I have taken one or more courses about ML", seven "I am familiar with the basic concepts", and one "I don't know very much about it". We recruited participants by word-of-mouth, and we rewarded them with a $25 USD Amazon gift card.

Task and Procedure

We asked participants to build an extractor for U.S. addresses, using a sampling dataset of 2,767 unlabeled text documents derived from web pages, which contained significant information about U.S. businesses.

We first gave participants a tutorial of PICL, and then we explained the task and what constitutes a U.S. address (Address contains Street, PO Box, City, State, Zip as sub-entities). This preliminary task took 20 minutes. Participants then had 20 minutes to create a U.S. address extractor. In practice, building a high-precision and recall extractor typically takes longer than 20

minutes, but sufficient to elicit the preliminary feedback we seek within the one-hour duration of the session.

We were interested in capturing information that allows us to observe how our designs and system augments a user's teaching capabilities. To have a fair evaluation of the participants' activities, we provided a full and well-structured entity hierarchy, and a rich set of 24 concept explanations, or features, sufficient to build a high precision and high recall extractor (e.g., an extractor with F-Score greater than 0.9 [7]). We also gave participants a written script that guided them on a document sampling strategy. To accelerate the speed at which good pre-labels become accessible and have the *Model Tracker* up-to-date with the latest labels, we instructed participants to retrain the model after every document submission. This setup put participants on an equal footing and enabled them to focus on the labeling and debugging tasks. We encouraged participants to think aloud while they were doing the task. In addition to keeping written notes from an observer and the system activity log, we recorded the screen and audio from each session for later analysis.

Measurements

We measured each participant's labeling and debugging activity as well as the performance of their information extraction model over time. We measured the labeling activity by the number of labeled documents and the numbers of tokens labeled as ADDRESS and non-ADDRESS. For our analysis we focused on the main concept, ADDRESSES, and not its constituents parts (e.g., STATE). We consider that a predicted address matches a label if they coincide exactly. Using the above measurements, we estimated the performance of an extraction model by computing precision, recall, and F-Score using a large test set of labeled documents containing 2,309 addresses. For this study, the F-Score value serves a direct estimation of the *efficacy* for a particular teaching session.

We analyzed the participants' labels against the ground truth labels for any mislabels that the participants introduced during their labeling and debugging activities. We categorized these labeling errors into ADDRESS and sub-ADDRESS errors. We further categorized ADDRESS errors into 1) missed errors (i.e., failed to label a valid Address) and 2) boundary errors (i.e., failed to include preceding or trailing words of an ADDRESS). We further categorized Sub-ADDRESS errors into 1) boundary errors (i.e., failed to include preceding or trailing word of an entity), 2) punctuation errors (i.e., included trailing punctuation word as part of an entity), and 3) incorrect entity (i.e., assigned the wrong entity to a range of words). We used the result of this error analysis to understand each participant's labeling behavior. Finally, we collected demographic information as well as qualitative feedback on participants' experience with the system in a post-study survey.

Results

Figure 7 provides a summary that quantifies our participants' labeling, reviewing activities, and model performance at the end of a 20 minutes teaching session. Complementary, figure 8 summarizes qualitative questions and answers related to labeling, debugging, and the general system experience. Participants did reasonably well with the extractors they were able to build.

[7] Prior to our study, we had a non-ML expert use our tool without a time constraint. This user was able to achieve an F-Score above 0.9 after 3 hours of teaching, starting with nearly identical initial conditions as our participants. F-Score is a common approach to combining the precision and recall of a system into a single number.

We observed that half of the participants were able to train an extractor with an F-Score greater than 0.6 by the end of the session.

Eight out of ten participants were satisfied with their overall experience with the system (Q2). In open response questions, they indicated that they liked: the "speed the system learns", that "after a while the system started giving the pre-labeled suggestions", "training a model and seeing it improve", seeing "disagreements marked in red".

During the study, we observed that participants varied in their dexterity in using a computer, the speed at which they understood how the system works, and their ability to scan a document for addresses. This influenced the 1) number of labels, 2) accuracy of labels, and 3) diversity of labels people produced. We believe these factors contribute to the variability in the performance of the models that the participants produced.

	Precision	Recall	F-Score	# Documents	# Addresses	# Errors
P1	0.7	0.57	0.63	12	10	0(0)
P2	0.75	**0.7**	**0.73**	19	**47**	1(0)
P3	0.85	0.38	0.53	11	35	7(7)
P4	0.85	0.64	**0.73**	**24**	40	4(0)
P5	0.27	0.23	0.25	9	22	7(0)
P6	0.84	0.53	0.65	23	42	**11**(6)
P7	0.68	0.56	0.61	8	**47**	0(0)
P8	0.75	0.44	0.55	11	17	3(1)
P9	**0.88**	0.29	0.44	16	33	**11**(11)
P10	0.57	0.5	0.53	6	5	0(0)
mean	0.7	0.5	0.6	13.9	29.8	4.4
sd	0.2	0.1	0.1	6.3	15.3	4.4

FIGURE 7. Summary of participants' labeling activity, label errors, and model performance at the end of a 20 minutes teaching session. Columns 2-4 indicate the precision, recall and F-Score of the final information extraction model. Columns 5-7 indicate the number of documents labeled, the number of ADDRESS **entities labeled, and the total number of** ADDRESS **errors with the number of missed** ADDRESS **entities errors in parentheses. The highest values for each column appears in bold.**

Observations and Insights

In this section, we present and discuss our quantitative and qualitative observations and put them in context with our first set of research questions. In particular, we distill our observations and insights into takeaway themes for practitioners interested in the design of ITEs.

RQ1.1 - What are unforeseen usability and design issues for IMT activities?

The importance of quantity and quality of labels. We did see evidence that the number of labels and the accuracy of the labels can contribute to the model performance. Participants who were able to submit sufficient addresses quickly were then able to have early access to good quality pre-labels.

Question	n/a	sd	d	n	a	sa
Q1: I was able to build an address extractor.	-	-	-	1	1	8
Q2: I had a satisfying experience using the system.	-	-	-	2	6	2
Q3: I found it helpful when the system provided a "pre-label" for a new example.	-	-	-	-	3	7
Q4: After I label an entity, I found it helpful when the system provided labels for other entities.	-	-	-	-	3	7
Q5: The Performance Tracker helped me understand the performance of my model.	-	-	-	1	3	6
Q6: The Performance Tracker clearly identified examples where the system disagreed with my labels.	-	-	-	-	4	6
Q7: The Performance Tracker made it easy to navigate to examples where the system disagreed with my labels.	1	-	-	-	2	7
Q8: The Performance Tracker made it clear that I was making progress in building an extractor.	-	-	-	-	5	5
Q9: It was easy to find where, in a document, the system disagreed with my labels.	-	-	-	-	5	5
Q10: After correcting a mislabeled example, the system generated pre-labels were useful in reducing the labeling effort.	3	-	-	-	2	5

FIGURE 8. Survey questions and associated counts for each answer. From left to right, columns correspond to: not applicable; strongly disagree; disagree; neutral; agree; and strongly agree.

This helped them increase the rate at which they could accurately label, as Figure 9 illustrates. Several participants, however, started with long documents (determined by the selected sampling strategy), which required them to scan for and potentially label many addresses (without the support of pre-labels) before they could submit. Encountering these types of documents early on delayed users from having access to useful pre-labels. Optimizing for efficient bootstrapping of a model is an important research direction for the future. The quality and quantity of the labels can be a factor of the interface design being accessible. While P5's low F-score can be attributed to a low number of labeled documents paired with a relatively high number of errors, we observed that those numbers were the result of the participant not being comfortable using the system's point-and-click interface. Providing accessible experiences is necessary for systems that are to be deployed at scale.

What the teacher sees is not necessarily what the system sees. We observed situations that a better design could have prevented. P10 spent the first eight minutes labeling a document with multiple addresses and retrained the model without submitting his labels, thus losing all these labels. P5 skipped a document that contained no addresses, missing the opportunity to submit meaningful negative labels. P9 missed six addresses because these addresses appeared below the fold of the document viewer's window. We need appropriate feedback to guide users to prevent errors or lost opportunities. Another source of problems is that in general, users focus mainly on positive labels (which are highlighted or underlined). This focus makes them not aware that when they submit a

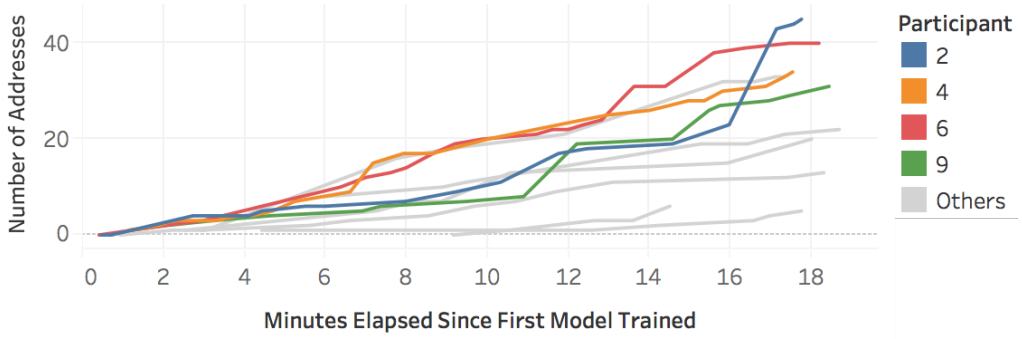

FIGURE 9. Cumulative number of *address* positive labels over time elapsed since the first trained model. The figure emphasizes four participants, where we see that the cumulative number of positive labels as a function of time. Lines follow a positive curvature trend, which can be interpreted as pre-labeling being helpful. For the other participants, the presence of pre-labels did not seem to harm their progress of labeling (i.e., while not as much as the four participants, the number of labels still increased).

document, they could be submitting false negative labels (i.e., areas with no visual decorations that do contain entities). Preventing the submission of mislabels as well as improving the experience of error discovery and correction are important parameters to consider when designing effective, usable, and safe ITEs.

Support concept evolution. Our analysis of labeling errors revealed interesting issues. 80% of the errors in sub-ADDRESS entities were due to the inclusion of punctuation characters. For some extraction applications, punctuation may not be critical. However, we believe it is important to be consistent with the labeling guidelines in both the training and testing phases of model training. An interesting direction to explore in future work is that of allowing teachers to participate in the creation and refinement of a labeling guideline. We believe that there is room for innovation in detecting and visualizing patterns of errors so that teachers can learn from and correct them.

Support access to diverse documents. We did not explicitly measure label diversity. However, we observed that some participants provided a diverse set of examples (e.g., an ADDRESS with and without PO BOX) even though we did not instruct them to do so. This diversity of examples typically improves recall.

RQ1.2 - How do machine-in-the-loop interventions support and affect IMT?

Pre-labels were helpful. We observed that pre-labels from a trained model help reduce the labeling time while model-assistance can shift the user task from explicit labeling to directed corrections. In particular, the number of produced labels was correlated with having access to good quality pre-labels, and as we observed, this leads to producing good models. The survey responses suggest that participants found pre-labels helpful and providing a benefit (Q3-4, and Q10). For example, "How did it even know that? It suddenly made it correct!" (P1), "I liked how the system started giving suggestions" (P2), or "The fact that the system took guesses was really important to me" (P8). This is despite the fact that a few subjects sometimes had difficulties with them. In par-

ticular, a participant would try to correct a mislabeled item by erasing an entity label after which the prediction label would add a different label that would then need to be erased again. While this behavior was not common, it highlights the need to craft the interaction of an assisted-labeling user experience carefully. While the few participants who encountered this behavior found it frustrating, they were nonetheless able to label these examples successfully. Also, users found pre-labels as a sign that the information extraction model was learning and getting better (e.g., "It felt like it learned really quickly with very few data points" (P3)).

Trust the machine, but verify. While pre-labels can be helpful, they introduce a validation task where teachers make sure that these predictions are right before submitting them. This task can lead users to apply effort making sure a pre-label is correct. Skipping this task can cause a teacher to submit mislabels. For example, P3 validated two pre-labeled addresses but failed to label many addresses between them. Pre-labels can have the undesired property that the user can trust them too much. It is important to carefully balance the model-assistance capabilities of mixed-initiative controls and teachers' over-reliance on the model.

Timely and interactive status for the model was useful. As indicated by the response to Q6 and Q7, participants also found the model tracker an effective means of identifying and pinpointing disagreements between submitted labels and model predictions. As with the presence of pre-labels, the tracker provided them with evidence that they were making progress in building their information extraction model. For example, by observing a histogram that separated true positives and negatives well or seeing a box change: "When I retrained the model the red went to gray - that helped me know that I was doing well" (P6). Finally, the document labeling control was helpful (Q9) for fixing and understanding disagreements between labels and predictions. The difference highlighting was effective, and pre-labels in this context also provided a benefit (Q10); "Red color was good for distinguishing errors" (P6), "I liked that the disagreements were red - made it easy to navigate" (P2). Interestingly, a few of the subjects did not get significant benefit from the Model Tracker or the Review Labeled-Example component. This was primarily due to the fact that they were extremely careful in labeling examples, so there were no mislabeled examples for the review model component to highlight; that is, they had no need for debugging during their session.

Limitations

In this study, we did not focus on tasks beyond labeling and debugging. It remains as part of our research road map to look in detail how PICL supports teachers with tasks such as defining and refining concept explanations, defining and refining a schema, or sampling documents.

Perhaps even more important is the need to understand how to guide teachers on how to teach. Our study gives evidence that non-ML experts can succeed in complex model training activities such as creating an entity extractor. That said, these users had at their disposal a rich set of 24 concept explanations (features). If they had not been given such a set, they would either have had to create them or would have needed to label many more documents. Understanding how to enable a teacher to choose their next tasks is critical. While there is some work on this topic for classification (Meek, Simard, & Zhu, 2016), it is unclear how to generalize to broader settings, even in the elemental case of building classifiers.

5.3. A study on guiding novice teachers

Our design probe underscored how the general IMT loop presents teachers with numerous decision-making points. Teachers choose elements to populate the training set, labeling them, and deciding when and how to fix prediction errors by selecting or creating semantically meaningful features influenced by what they have seen. As this teaching process unfolds, different teachers can take different actions and paths to teach the same model. This is akin to the many ways there are for different programmers to implement a function.

Part of the appeal of IMT is that teachers only require expertise about a subject-domain, and no knowledge about the details, such as type and parameters, of the underlying learning algorithm. This set-up leads to a process requiring a softer skillset than traditional ones that require ML knowledge.

In this section, we summarize and discuss the results and lessons learned from a study (Wall et al., 2019), where we explored and showed how we can guide teachers with minimal or no ML or IMT knowledge through the teaching process to build classifiers comparable to those built by seasoned machine teachers. In particular, we focused on identifying expert teaching patterns and studying how to pass them to novice end-users so that they can be effective teachers early on. Through this study, we looked to answers the following second set of research questions:

- RQ2.1 - Can novice teachers be quickly on-boarded and use an ITE behaving similarly to teachers with more experience?
- RQ2.2 - Can we codify expert teaching patterns and use them to assist novice teachers?

Observing Expert Teachers

While there is no universal best way to implement or code a function (in a particular programming language), there are best practices, patterns, and principles developed over the years that make the process of coding efficient and manageable, especially in the context of complexity. IMT, as an area of study, is in its at the early stages, and current best practices are locked in the experts developing the field and applying it to the process of building ML models. We aim at codifying this expertise to serve as the basis of expert teaching patterns that can be of use to novice or inexperienced teachers.

To achieve our goal, we had access to a group of seasoned machine teachers with more than four years of experience applying IMT principles and performed a qualitative study observing them building ML models using a variation of PICL we called MATE (for machine teaching environment). In this variation, we used a model tracker where unit boxes were colored green, red, or blue if a teacher labeled a document positive, negative, or undecided, respectively. We also simplified the choices and language of the sampling strategy control to be more approachable for novice teachers. Lastly, MATE included a notification system that we used to provide teaching guidance to its users. Figure 10 illustrates.

During the studies, we observed these teachers teach a binary classifier about a general-domain topic using a sampling set of 90,000 documents scraped from recent news articles and blog posts from a data-aggregation service (Webhose, 2020). In addition to our observations, we also elicited direct feedback and rationale as to each of the seasoned teacher's actions. We refer readers to (Wall et al., 2019) for more details about our study design, set up, and other results.

FIGURE 10. MATE's main screen. (1) model tracker. (2) sampling control. (3) labeling control. (4) featuring interface. (5) notifications.

Phases of teaching

From our observations we identified four high-level, non-overlapping phases or states that teaching experts went through during their teaching sessions: *Cold Start*, *Boundary*, *Challenge*, and *Testing*. Figure 11 illustrates.

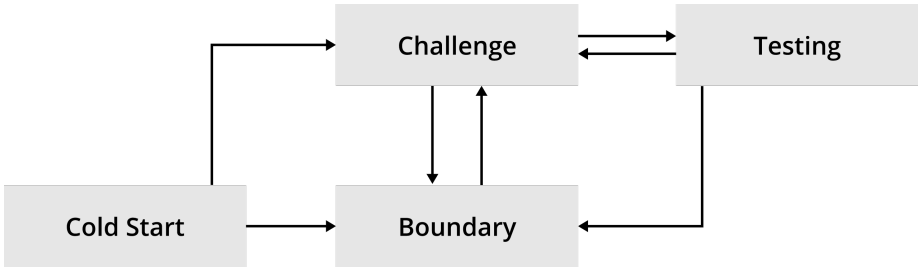

FIGURE 11. The four stages during an IMT session. Teachers begin at cold start, once the model-in-training starts improving, the teacher can transition to a boundary, or challenge phase depending on how well the model is separating the training set. If the model seems to be separating the training set well, the teacher might decide to go to a testing phase, where they review the model as a whole.

The Cold Start phase occurs at the very beginning of a teaching session, where there are none or barely any labels or features given to the learning algorithm. During this phase, teachers create

features to roughly capture the positive class, and they provide a minimum amount of positive and negative labels. One can transition from this phase to Boundary or Challenge.

The Boundary phase is defined by a lack of a clear boundary separating positive and negative labels in the training set. During this phase, teachers look into improving the separation between the positive and negative labels. Teachers rely on foresight and the system's uncertainty sampler to find and label documents that the model does not predict with confidence. Teachers also select labeled examples near the decision boundary and refine the system's collection of concept explanations to either correct blindness errors or improve the prediction score on a document. One can transition from this phase to the Challenge phase.

The Challenge phase takes place when there is a clear boundary separating positive and negative labels over the training set. During this phase, teachers look for ways to challenge the model (i.e., find interesting examples where the model is likely to make a bad prediction). To do this, teachers use foresight and a variety of sampling strategies to find and label diverse and challenging data. One can transition from this phase to Testing or Boundary.

The Testing phase occurs when a teacher explicitly evaluates the quality of the model on unseen documents. This phase can occur periodically throughout the teaching process, most commonly transitioning from the Challenge Phase. Experts test their model by looking at the documents fetched using the Predicted Positives sampling strategy, and counting how many predictions out of a batch are correct; no new labels or features are created in this phase. The higher the number of correct predictions, the more confident the teacher feels of the model's behavior on unseen data. One can transition from this phase to Boundary or Challenge.

Expert Teaching Patterns and Guidance

As we synthesized our observations from seeing experts' teaching strategies into actionable guidance, we sought to characterize prescriptive actions as well as conditions for when such actions are useful. Figure 12 summarizes our observations translated into actionable guidance given a particular teaching state.

As our goal is to study how to guide or help novice teachers, it was important to consider not only what to say, but also when to say it. This does not only refers to what states an action should follow but also that is the proper moment during a teacher's interaction that the ITE should present a (teaching) suggestion to the teacher. We follow a naïve mediated approach where we pick moments between tasks (a strategy supported by work like Cheng, Teevan, Iqbal, & Bernstein, 2015), or situations of apparent inactivity. There are several such moments during a teaching session:

1. after submitting a label,
2. after closing the review of a labeled document,
3. (while reading a new document) after 30, or 60 seconds of inactivity[8],
4. after getting a document with the same sampler without labeling 10 times in a row,
5. after creating/editing a feature, and
6. after creating/editing features for 5 consecutive minutes.

The mechanics of our teaching guidance subsystem with the ITE work as follows: whenever an interruptible moment happens, the system chooses the proper guidance to show based on the

[8]A proxy for reading: hovering over, or scrolling a document.

system's current state. If there is ambiguity as to what guidance to provide, the system picks one at random. The chance of a particular guidance being picked significantly decreases if the guidance has been seen (and acknowledged by the teacher) within the past 10 minutes.

Our goal of adding a notification system to the ITE is to provide guidance throughout the teaching process to novice teachers, including sampling, featuring, and labeling guidance. When synthesizing our observations from experts' teaching strategies into actionable guidance, we sought to characterize prescriptive actions as well as conditions for when such actions are useful. Figure 12 summarizes the guiding notifications we produced from our observations.

To study the effects of applying this teaching guidance, we performed a user study measuring the differences between the two groups of novice teachers engaged in the creation of a binary classifier. Of these groups, one had access to the patterns and system guidance described above, and the other did not.

The Effects of Guiding Teaching Patterns

Quantitative Results

After teaching sessions lasting 45 minutes, we did not observe a significant effect of the guiding intervention on the resulting model's F1-Score, number of produced labels, number of explanations, or total number of keywords used across explanations. What was interesting about this result was the fact that the values for these measures were not too far from the average numbers produced by three seasoned teachers. This constitutes encouraging evidence suggesting that with minimal training, novices teachers can be highly functional machine ones.

This result also raises the question of methodology, in particular, the duration of the teaching sessions. It is possible that the presence of any effects from having a guidance system would be observable under more realistic teaching times. In other words, several sessions that add to three or more hours. Further research in this area needs to look into experimental designs that capture teaching parameters closer to real-world ones.

Nevertheless, teachers operating under the guided condition produced better models with higher F1-scores, more labels, and a similar number of features, but with fewer keywords. In concert, these are indicators of efficient teaching.

Qualitative Results

Our study gathered qualitative feedback using a variation of NASA's TLX survey, which we used to measure responses along five qualitative dimensions: perceived success level, frustration, mental demand, pace, and effort. We also did not observe a significant effect of the guiding intervention on the different qualitative scores. Like in the case above, this could be explained by the teaching sessions being not long enough, or the number of participants (twelve in each condition) not able to capture significant effects.

Despite these results, the surveys showed that guided teachers expressed generally lower frustration, less mental demand, and consistent effort ratings, in contrast with a high variance within the non-guided teachers. These results shed a positive light on having a teaching guiding system, as real-world teaching sessions are significantly longer than 45-minutes.

Teachers that had expert patterns revealed to them as guidance rated them as helpful and timely, with lower levels of confusion. Seen them as a whole, these and the above results pro-

Name	Guidance	Heuristic
Cold Start: Positive	Use Keyword Search to find and label 10 positive examples.	At the start of the teaching process.
Cold Start: Negative	Use Random Sampler to find and label 10 negative examples.	After 10 positive examples have been labeled.
Boundary Phase	Use Uncertainty to find and label 10 examples.	Predictions are not well separated.
Challenge Phase	Change samplers often and find diverse documents to label.	Predictions are well separated.
Test Phase	Use Predicted Positives to see how accurately the model predicts 10 unseen documents.	After every 50 labels; predictions are well separated.
Tip: Create Feature	Create a feature that explains a positive label.	Reviewing a false negative.
Tip: Test Feature	Use Explanation-Based to see if a feature captures documents in the context the teacher expects.	After creating or modifying a feature, during Boundary Phase or Challenge Phase. Do not show during Cold Start Phase.
Tip: Possible Errors	Use Possible Errors to find documents the model is likely predicting incorrectly.	After Cold Start Phase; predictions are well-separated.
Tip: Review Errors	Review training errors.	Predictions are not well-separated; the number of errors is high.
Tip: Feature Suggestion	Check if the system suggests any sensible feature.	The tool suggests features; number of errors is high; predictions are not well-separated.
Tip: Decide Later	Use *decide later* if the teacher is not sure how to label; or they cannot ideate a feature to explain a label.	Reviewing a document for ≥ 30 seconds, or revisiting the same document ≥ 3 times.
Tip: Negative Feature	Create a feature to describe why a label is negative.	Reviewing a false positive.

FIGURE 12. The guidance provided to participants along with heuristics for when notifications can be dispatched. We say predictions are *well separated* when $\leq$ 10% of prediction scores are between 0.3-0.7. We say the number of errors is *high* when $\geq$ 20% of documents are predicted incorrectly in the model.

vide arguments in favor of using and revealing the teaching patterns as guidance in a progressive disclosure way.

Observations and Next Steps

The study we reported on (Wall et al., 2019) only begins to explore both the space of expert teaching patterns and the design space of solutions to present them to machine teachers. Further work in this space includes synthesizing expert patterns beyond binary classification (e.g., extraction tasks). This study helps to provide some answers to our second set of research questions.

RQ2.1 - Can novice teachers be quickly on-boarded and use an ITE behaving similarly to teachers with more experience?

Our results and observations showed that given a short amount of training in ML and IMT through three 5-minute videos, novice teachers are able to construct models of similar quality to those built by seasoned machine teachers.

RQ2.2 - Can we codify expert teaching patterns and use them to assist novice teachers?

Our results and observations let us identify the teaching phases of IMT systems. They also showed that expert teachers use patterns that make them progress through the IMT loop. We also saw that guiding novice teachers with the aforementioned expert patterns helps them behave like expert teachers and build models with less effort than novice end-users without such guidance.

There are other interesting insights from this study. While we studied a push style of notification, other styles exist. Pull guidance in the form of a teacher explicitly asking for assistance is a reasonable possibility, as well as approaches where there is a dedicated area that always displays suggestions as to what is helpful to do. Some types of guidance are more nuanced than a specific action, such as *add a negative label*. For example, while it is clear about the type to action to take to correct a false positive prediction, that is to provide a new or refined concept explanation, it is up to the teacher's insight to discern and articulate it. Even under the presence of mixed-initiative tools that suggest potential choices, the final semantic decision falling on the teacher is not obvious.

One main observation during the study is that teachers struggle with the concept of negatives in general. This observation aligns with one made by (Yang, Suh, Chen, & Ramos, 2018) regarding the optimistic mentality non-experts use when building ML models. In the context of our study, this manifested in two ways. First, it was not obvious for some non-guided teachers when negative examples needed to be added into the training set, or how many. Second, almost consistently, teachers on both the guided and non-guided conditions asked a question of the form "how do I articulate a negative explanation?", in the context of attempting to correct a false positive error. Guided teachers were quickly helped by the system that suggested that to fix a false positive, it was better to think what the example *actually is* as opposed to what is not. Our ITE's teaching language philosophy does not distinguish between a positive or a negative explanation for a concept. It purposely only allows for teachers to articulate concept explanations that will then be used by the learning algorithm to produce a model. While this design choice gives teachers an expressive way to teach, it is reasonable to think about the different ways one can support teachers' natural ways of expressing themselves. Lastly, patterns and tooling that helps teachers discover *effective* concepts, or their decomposition is a challenging, yet full of reward area of future work. In the next section, we discuss research directions for IMT in general.

6. RESEARCH OUTLOOK

In contrast to the study of machine learning, the study of interactive machine teaching focuses on the role of a subject matter expert and their interaction with a learning algorithm rather than focusing on the algorithms by which a machine learns. By focusing on interactive machine teaching, we explore a class of IML processes that have the potential to broaden the access to and simplify the creation of machine-learned models. Our work underscores this potential while making it clear that there are important research challenges at the intersection of HCI and ML yet to be addressed. In this section, we review some of these challenges.

6.1. Leveraging rich human knowledge and capabilities

A fundamental research question in IMT is to enable a teacher to express themselves and leverage that knowledge, expertise, and human capabilities in building intelligent systems. The set of research challenges related to this goal is rich, diverse, and growing. This growth is driven by an increasing desire to build intelligent systems and to apply them to a growing diversity of application domains. To succeed in this endeavor will require interdisciplinary collaboration and expertise. A grand challenge for IMT research is exploring the range, diversity, and types of human knowledge, and understand how learning systems can leverage this knowledge. The magnitude of this challenge becomes clear when one also considers the diversity of potential scenarios like process control (e.g., robotics, manufacturing), document, image understanding, question answering, and information extraction.

6.2. Teaching languages

Another fundamental area of future research is the study and development of different teaching languages. The PICL system targets the classification and extraction for one-dimensional text though concept explanations. Other platforms like Bonsai (Bonsai, 2020a) target training reinforcement learning systems through the specialized Inkling language. How will current teaching languages need to change or will new ones need to be created to allow teachers to express concepts and relationships in higher-dimensional documents such as web-pages, images or, videos? Teaching languages for control scenarios like Inkling (Bonsai, 2020b) are currently very different than a language for one-dimensional text. Can they or should they be unified? What types of knowledge beyond labels and concepts can be effectively conveyed and leveraged?

6.3. End-user machine learning and user experience design

PICL provides teachers with one modality of teaching, within a specific *interaction paradigm* (e.g., Windows Icons Menus Pointer – WIMP). As other emerging interaction paradigms that go beyond (large) screens and pointing devices take center stage, we inquire how teaching can take place in these other paradigms, such as voice dialog systems. Are any of these interaction paradigms better suited for large sets of concepts and documents? Furthermore, can we apply the insights from our studies and apply them to other documents besides text such as images? Are particular human capabilities better supported by different interaction modalities? Something to consider about the

design of IMT interactions relates to the cardinality of teachers and learners. While we have discussed IMT in terms of a single teacher and learner, it is possible to think about scenarios where one leverages the capabilities of multiple teachers with a diverse set of skills and expertise. Are there principles about how we can enable multiple people to work effectively together? Or is there a mythical-man-month analogy (Brooks, 1975) for teaching? One of the ways that multiple software developers effectively work together is by creating packages and libraries. Will the development of teaching languages lead to the creation of knowledge libraries that are shared and used by teachers in achieving their teaching goals?

6.4. Machine-in-the-loop teaching

Seen as a process, IMT is subject to some form of control. As a teaching session evolves, does it make sense for the controller role to alternate between the teacher and the learner? If a teacher relinquishes control, how would we best support them in recovering it? What are the proper metrics or practices that can help teachers know when they are done teaching? The imperfect nature of intelligent systems and the iterative nature of teaching the ways people do leads to a variety of challenges. How do we design system interventions that do not replace, but *complement* or *amplify* human capabilities? Can novice teachers be quickly on-boarded and use an ITE behaving similarly to teachers with more experience? How do we communicate and leverage information about the uncertainty in a prediction or the reliability of uncertain knowledge? Our observations showed that the design of system interventions can significantly impact a teaching session. Trusting a system's predictions too early, or not visualizing them in understandable, effective ways can generate unwanted mislabels that deteriorate a model's performance.

6.5. Interactive machine teaching literacy and teaching teachers

We have discussed decomposition as a tool for teaching. We know from computer education that leveraging decomposition is not an innate skill but rather one that needs to be learned. How can we best teach teachers skills like decomposition? What are the other skills that teachers need to be taught? What are the proper ways to pass on these skills? Our research has only begun to shed light on the topic, and there are opportunities for exploring the connections and potential synergies between IMT and the study of human teaching (pedagogy).

6.6. Interactive machine teaching as programming

While IML has been used to develop many intelligent systems, we note that the use of machine learning in these systems requires significant expertise in machine learning and comes with significant costs (Sculley et al., 2015; Simard et al., 2017). IMT has the potential to enable machine learning to be more accessible and effectively used in the creating of intelligent software systems while reducing these costs.

One approach that we have found to be particularly useful in exploring research opportunities is to consider the analogy between IMT and programming. The teaching process (e.g., as described by the IMT loop in Figure 1) has striking similarities with the process by which a software engineer creates a program (see Simard et al., 2017 for a more extended discussion). The end goal of creating a functional artifact is the same. This suggests that the goals of IMT are similar to those of end-user-programming: to enable non-programmers to create functions or programs. The primary

distinguishing characteristic is that IMT aims to leverage the subject-matter expertise of the end-user and leverages a learning machine to compile this expertise into functional artifacts. In the past few decades, there has been an explosion of tools and processes aimed at increasing programming productivity. These include the development of high-level programming languages, innovations in integrated development environments, and the creation of development processes. Figure 13 presents a mapping of many of these tools and concepts to IMT. Each of the breakthroughs that have advanced programmer productivity is likely to have an analogous breakthrough in IMT. Thus, the history of advancements in software engineer is likely to provide clues to what is yet to come in the development of IMT.

Programming Tools and Concepts	Interactive Machine Teaching
Programming languages constructs (variables, classes, functions, etc.)	Teaching languages constructs (features, schemas, demonstrations, etc.)
Frameworks, Packages, etc.	Model zoos, Feature repositories, ImageNet, word2vec, etc.
Compiler	Learning algorithms (neural networks, SVMs)
Integrated development environments (syntax highlighting, compiling, debugging, command completion, refactoring)	Integrated teaching environments (feature highlighting, training, sampling, debugging, etc.)
Programming expertise (decomposition, patterns, architecture)	Teaching expertise (decomposition, curriculum learning, etc.)
Version control for source code	Version control for data, labels, features, schema, etc.
Development processes (specifications, unit testing, deployment, monitoring, etc.)	Teaching processes (data collection, testing, publishing, etc.)

FIGURE 13. Mapping between programming tools and concepts and interactive machine teaching.

7. CONCLUSION

In this article, we describe interactive machine teaching, an approach to creating semantic, debuggable machine-learned models. This approach leverages people's teaching capabilities and their subject-matter expertise in an iterative process. We articulate a set of intrinsic human capabilities related to teaching, and how IMT systems might leverage them. Through the implementation and study of an integrated teaching environment, we share observations and insights as to how people teach and can be guided to teach an ML model.

We believe that advancements in research on interactive machine teaching will help to em-

power people with access to new, personalized, and abundant information processing capabilities. By making ML model creation accessible to subject-matter experts, that only need the capacity to teach, IMT can bring the ability to create models to scenarios where ML expertise is scarce, costly, or perceived as impractical. For example, a legal clerk could filter hundreds of documents about a particular matter by teaching a system what the concepts of interest are. Similarly, a person can adapt the behavior of their smart-home based on teaching a system what the important events from heterogeneous sensors, contextual information of the house, demonstrations, and foresight rules are. As most people already possess the mindset and ability to teach an ML model, one can speculate that the number of available ML models for people to use for either complete tasks or build other models will be plentiful. We also speculate that this explosion in the number of ML models will lead to the formation of repositories, where people publish and consume ML models. Besides providing a library of functionality for everybody, these repositories can accelerate the creation of ML models where a teacher leverages other sub-models as semantic features. Furthermore, it is not far-fetched to see these repositories evolving into marketplaces that can fuel new economic opportunities for businesses and individuals alike.

We are excited by the research challenges provided by interactive machine teaching. As a field of study, it stands in contrast to the field to ML, where instead of focusing on extracting knowledge from data, it focuses on extracting knowledge from people. This shift in perspective is natural as people have the goal of building ML models. Analogous to the rapid advancement of programming productivity, we expect to see rapid progress in interactive machine teaching in the years to come.

REFERENCES

Amazon. (2020). *Amazon Sagemaker: Machine learning for every developer and data scientist.* Retrieved 2020-01-31, from https://aws.amazon.com/sagemaker

Amershi, S. (2011). *Designing for Effective End-User Interaction with Machine Learning* (Doctoral dissertation). Retrieved from https://www.microsoft.com/en-us/research/publication/designing-effective-end-user-interaction-machine-learning/

Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. *AI Magazine*, *35*(4), 105–120.

Amershi, S., Chickering, M., Drucker, S. M., Lee, B., Simard, P., & Suh, J. (2015). Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the CHI 2015 Conference on Human Factors in Computing Systems* (pp. 337–346). New York: ACM.

Amsel, A. (1989). *Behaviorism, Neobehaviorism, and Cognitivism in Learning Theory: Historical and Contemporary Perspectives*. Hillsdale, New Jersey, USA: Lawrence Erlbaum.

Apple. (2020). *Turi Create simplifies the development of custom machine learning models.* Retrieved 2020-01-31, from https://github.com/apple/turicreate

Au, T. C. (2018). Random forests, decision trees, and categorical predictors: the absent levels problem. *The Journal of Machine Learning Research*, *19*(1), 1737–1766.

Azure, M. (2020). *Azure ML Studio.* Retrieved 2020-01-31, from https://studio.azureml.net/

Bates, M. J. (1998). Indexing and access for digital libraries and the internet: Human, database, and domain factors. *Journal of the American Society for Information Science*, *49*(13), 1185-1205.

Ben-Ari, M. (1998). Constructivism in computer science education. In *Proceedings of the SIGCSE 1998 Technical Symposium on Computer Science Education* (pp. 257–261). New York: ACM.

Bonsai. (2020a). *Bonsai: BRAINs for Autonomous Systems.* Retrieved 2020-01-31, from https://bons.ai

Bonsai. (2020b). *Inkling: Bonsai's special-purpose programming language for training AI.* Retrieved 2020-01-31, from https://docs.bons.ai/guides/inkling2-guide.html

Brooks, F. P., Jr. (1975). The mythical man-month. In *Proceedings of the International Conference on Reliable Software* (pp. 193–). New York.

Cakmak, M., & Thomaz, A. L. (2014). Eliciting good teaching from humans for machine learners. *Artificial Intelligence*, *217*, 198 - 215.

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Information Visualization*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 2015 SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1721–1730). New York: ACM.

Cheng, J., Teevan, J., Iqbal, S. T., & Bernstein, M. S. (2015). Break it down: A comparison of macro- and microtasks. In *Proceedings of the CHI 2015 Conference on Human Factors in Computing Systems* (pp. 4061–4064). New York: ACM.

Chernova, S., & Thomaz, A. L. (2014). *Robot Learning from Human Teachers* (Vol. 8) (No. 3). Hillsdale, New Jersey, USA: Morgan & Claypool Publishers.

Culotta, A., & McCallum, A. (2004). Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers* (p. 109–112). USA: Association for Computational Linguistics.

Doshi-Velez, F., Kortz, M., Budish, R., Bavitz, C., Gershman, S., O'Brien, D., … Wood, A. (2017). Accountability of AI under the law: The role of explanation. *CoRR*, *abs/1711.01134*.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Dudley, J. J., & Kristensson, P. O. (2018). A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *8*(2), 8.

Fails, J. A., & Olsen, D. R., Jr. (2003). Interactive machine learning. In *Proceedings of the IUI 2003 International Conference on Intelligent User Interfaces* (pp. 39–45). New York: ACM.

Felix, C., Dasgupta, A., & Bertini, E. (2018). The exploratory labeling assistant: Mixed-initiative label curation with large document collections. In *Proceedings of the UIST 2018 Symposium on User Interface Software and Technology* (pp. 153–164). New York: ACM.

Fiebrink, R., & Cook, P. R. (2010). The Wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of the ISMIR 2010 International Society for Music Information Retrieval Conference.*

Fogarty, J., Tan, D., Kapoor, A., & Winder, S. (2008). Cueflik: interactive concept learning in image search. In *Proceedings of the CHI 2008 Conference on Human Factors in Computing Systems* (pp. 29–38). New York: ACM.

Fusi, N., Sheth, R., & Elibol, M. (2018). Probabilistic matrix factorization for automated machine learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31* (pp. 3348–3357). Curran Associates, Inc.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *Proceedings of the DSAA 2018 IEEE International Conference on Data Science and Advanced Analytics* (pp. 80–89).

Goodman, B., & Flaxman, S. (2017). European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, *38*(3), 50–57.

Google. (2020a). *AI Platform: Create your AI applications once, then run them easily on both GCP and on-premises.* Retrieved 2020-01-31, from `https://cloud.google.com/ai-platform`

Google. (2020b). *Teachable Machine: Train a computer to recognize your own images, sounds, & poses.* Retrieved 2020-01-31, from `https://teachablemachine.withgoogle.com/`

Google. (2020c). *Tensorflow: An end-to-end open source machine learning platform.* Retrieved 2020-01-31, from `https://www.tensorflow.org/`

Hohman, F., Head, A., Caruana, R., DeLine, R., & Drucker, S. M. (2019). Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the CHI 2019 Conference on Human Factors in Computing Systems* (pp. 579:1–579:13). New York: ACM.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 159–166). New York: ACM.

Hutter, F., Lücke, J., & Schmidt-Thieme, L. (2015). Beyond manual tuning of hyperparameters. *KI-Künstliche Intelligenz*, *29*(4), 329–337.

IBM. (2020). *Watson Studio: Simplify and scale data science to predict and optimize your business outcomes.* Retrieved 2020-01-31, from `https://www.ibm.com/cloud/watson-studio`

Jandot, C., Simard, P., Chickering, M., Grangier, D., & Suh, J. (2016). *Interactive Semantic Featuring for Text Classification.* Retrieved from https://arxiv.org/abs/1606.07545v1

Johns, E., Mac Aodha, O., & Brostow, G. J. (2015). Becoming the expert - interactive multi-class machine teaching. In *Proceedings of the CVPR 2015 Conference on Computer Vision and Pattern Recognition.*

Jupyter, P. (2020). *Jupyter Notebooks.* Retrieved 2020-01-31, from https://jupyter.org/

Kane, D. M., Lovett, S., Moran, S., & Zhang, J. (2017). Active classification with comparison queries. *CoRR*, *abs/1704.03564*. Retrieved from http://arxiv.org/abs/1704.03564

Kanter, J. M., & Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. In *Proceedings of the DSAA 2015 International Conference on Data Science and Advanced Analytics* (pp. 1–10).

Krening, S. (2019). *Humans Teaching Intelligent Agents with Verbal Instruction* (Doctoral dissertation, Georgia Institute of Technology). Retrieved from https://smartech.gatech.edu/handle/1853/61232

Krening, S., & Feigh, K. M. (2018, October). Interaction algorithm effect on human experience with reinforcement learning. *Journal of Human-Robot Interaction*, *7*(2).

Krening, S., & Feigh, K. M. (2019). Effect of interaction design on the human experience with interactive reinforcement learning. In *Proceedings of the DIS 2019 Designing Interactive Systems Conference* (p. 1089–1100). New York: ACM.

Kulesza, T., Amershi, S., Caruana, R., Fisher, D., & Charles, D. (2014). Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the CHI 2014 Conference on Human Factors in Computing Systems* (pp. 3075–3084). New York: ACM.

Kulesza, T., Burnett, M., Wong, W.-K., & Stumpf, S. (2015). Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the IUI 2015 International Conference on Intelligent User Interfaces* (pp. 126–137). New York: ACM.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the SIGKDD 2016 International Conference on Knowledge Discovery and Data Mining* (pp. 1675–1684). New York: ACM.

Lane, D. (2020). *Machine Learning for Kids.* Retrieved 2020-01-31, from https://machinelearningforkids.co.uk

Lipton, Z. C. (2018, June). The mythos of model interpretability. *Queue*, *16*(3), 30:31–30:57.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30* (pp. 4765–4774). Curran Associates, Inc.

Matas, M., Menges, A., & Beissinger, M. (2020). *Lobe: Deep Learning Made Simple.* Retrieved 2020-01-31, from https://lobe.ai/

Maulsby, D., Greenberg, S., & Mander, R. (1993). Prototyping an intelligent agent through wizard of oz. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 277–284). New York: ACM.

Meek, C. (2016). A characterization of prediction errors. *CoRR*, *abs/1611.05955*. Retrieved from http://arxiv.org/abs/1611.05955

Meek, C., Simard, P., & Zhu, X. (2016). Analysis of a design pattern for teaching with features and labels. *CoRR*, *abs/1611.05950*. Retrieved from http://arxiv.org/abs/1611.05950

Microsoft. (2020). *LUIS: Language Understanding.* Retrieved 2020-01-31, from https://www.luis.ai/

Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. In *Proceedings of*

the FAT 2019 Conference on Fairness, Accountability, and Transparency (pp. 279–288). New York: ACM.

Nowak, M., Castellini, C., & Massironi, C. (2018). Applying radical constructivism to machine learning: A pilot study in assistive robotics. *Constructivist Foundations*, *13*(2), 250-262.

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference* (pp. 485–492).

Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Kidd, L. C., & Moore, J. H. (2016). Automating biomedical data science through tree-based pipeline optimization. In *Proceedings of the 2016 European Conference on the Applications of Evolutionary Computation* (pp. 123–137).

Patel, K., Bancroft, N., Drucker, S. M., Fogarty, J., Ko, A. J., & Landay, J. (2010). Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the UIST 2010 symposium on User interface software and technology* (pp. 37–46). New York: ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pérez, F., & Granger, B. E. (2007, May). IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, *9*(3), 21–29.

Phillips, D., & Soltis, J. (2015). *Perspectives on Learning, 5th Edition*. New York, NY, USA: Teachers College Press.

Pirolli, P., & Card, S. (2005). The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of the 2005 International Conference on Intelligence Analysis* (Vol. 5). McLean, VA, USA.

Prodigy. (2020). *Prodigy: Radically efficient machine teaching.* Retrieved 2020-01-31, from `https://prodi.gy/`

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2019, Jul 15). Snorkel: rapid training data creation with weak supervision. *The VLDB Journal*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. (2009, November). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the SIGKDD 2016 International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). New York: ACM.

Ritter, A., & Basu, S. (2009). Learning to generalize for complex selection tasks. In *Proceedings of the IUI 2009 International Conference on Intelligent User Interfaces* (pp. 167–176). New York: ACM.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206.

Sarkar, A. (2016). Constructivist design for interactive machine learning. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 1467–1475). New York: ACM.

Scarr, J., Cockburn, A., & Gutwin, C. (2013, December). Supporting and exploiting spatial memory in user interfaces. *Foundations and Trends in Human-Computer Interaction*, *6*(1), 1–84.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... Dennison, D. (2015).

Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems 28* (pp. 2503–2511). San Jose, California, USA: Curran Associates, Inc.

Settles, B. (2012). *Active Learning*. San Rafael, CA, USA: Morgan & Claypool Publishers.

Simard, P. Y., Amershi, S., Chickering, D. M., Pelton, A. E., Ghorashi, S., Meek, C., ... Wernsing, J. (2017). Machine teaching: A new paradigm for building machine learning systems. *arXiv preprint arXiv:1707.06742*.

Spacy. (2020). *Spacy: Industrial-Strength Natural Language Processing.* Retrieved 2020-01-31, from `https://spacy.io/`

Suh, J., Ghorashi, S., Ramos, G., Chen, N.-C., Drucker, S., Verwey, J., & Simard, P. (2019, August). Anchorviz: Facilitating semantic data exploration and concept discovery for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems*, *10*(1).

Sutton, R., & Barto, A. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press.

Ustun, B., & Rudin, C. (2017). Optimized risk scores. In *Proceedings of the SIGKDD 2017 International Conference on Knowledge Discovery and Data Mining* (pp. 1125–1134). New York: ACM.

Valdes, G., Luna, J. M., Eaton, E., Simone II, C. B., Ungar, L. H., & Solberg, T. D. (2016). Mediboost: a patient stratification tool for interpretable decision making in the era of precision medicine. *Scientific reports*, *6*, 37854.

Wadsworth, B. (1996). *Piaget's Theory of Cognitive and Affective Development: Foundations of Constructivism*. London, UK: Longman.

Wall, E., Ghorashi, S., & Ramos, G. (2019). Using expert patterns in assisted interactive machine learning: A study in machine teaching. In *Proceedings of the INTERACT 2019 Human-Computer Interaction Conference* (pp. 578–599). Paphos, Cyprus.

Wallace, B. C., Small, K., Brodley, C. E., Lau, J., & Trikalinos, T. A. (2012). Deploying an interactive machine learning system in an evidence-based practice center: Abstrackr. In *Proceedings of the SIGHIT 2012 International Health Informatics Symposium* (pp. 819–824). New York: ACM.

Ware, M., Frank, E., Holmes, G., Hall, M., & Witten, I. H. (2001). Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, *55*(3), 281 - 292.

Webhose. (2020). *Webhose.io: Tap Into Web Content at Scale.* Retrieved 2020-01-31, from `http://webhose.io`

Weld, D. S., & Bansal, G. (2019, May). The challenge of crafting intelligible intelligence. *Communications of the ACM*, *62*(6), 70–79.

Xu, Y., Zhang, H., Miller, K., Singh, A., & Dubrawski, A. (2017). Noise-tolerant interactive learning using pairwise comparisons. In *Advances in Neural Information Processing Systems 30* (pp. 2431–2440). Curran Associates, Inc.

Yang, Q., Suh, J., Chen, N.-C., & Ramos, G. (2018). Grounding interactive machine learning tool design in how non-experts actually build models. In *Proceedings of the DIS 2018 Designing Interactive Systems Conference* (p. 573–584). New York: ACM.

Zhu, X. (2015). Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI 2015 Conference on Artificial Intelligence.* Austin, Texas, USA.

Zhu, X., Singla, A., Zilles, S., & Rafferty, A. N. (2018). An overview of machine teaching. *CoRR*, *abs/1801.05927*. Retrieved from `http://arxiv.org/abs/1801.05927`