BEHAVIOR-DRIVEN AI DEVELOPMENT

Ángel Alexander Cabrera

CMU-HCII-24-101
April 2024

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

The research reported here was supported, in whole or in part, by an Amazon award, a Mozilla Technology Fund grant, a National Science Foundation grant under No. IIS-2040942, and the National Science Foundation Graduate Research Fellowship Program under grant No. DGE-1745016. All views expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

KEYWORDS

Machine learning evaluation, AI evaluation, failure analysis, behavioral analysis, sense-making, human-AI collaboration, visualization, crowdsourcing, machine learning, artificial intelligence.

ABSTRACT

AI systems are being deployed in many real-world applications, from self-driving cars to customer service chatbots. When a person interacts with an AI system, they develop a complex *mental model* of how the system behaves that they use to inform their interaction with the AI. Should I override the AI prediction? Should I collect more training data? Traditionally, aggregate metrics such as accuracy are calculated on a hold-out test set to measure a model's overall performance. This singular metric is often insufficient to develop mental models that capture important AI behaviors, such as potential biases or safety concerns.

This thesis proposes **behavior-driven AI development** (BDAI), a philosophy that centers AI development on identifying, quantifying, and communicating the numerous *behaviors* a model can show. By focusing on a model's behaviors instead of aggregate metrics, developers can focus on creating responsible AI systems that best fulfill end-user needs. BDAI is central to creating AI systems, informing how a model should be updated, and deploying AI, informing how people should interact with a model. In this thesis, I describe empirical and system-building work that formally defines BDAI and shows how it can be applied to improve real-world AI systems.

In the first half of the thesis, I present a series of interviews, a theoretical framework, and a user study that describe the core principles of BDAI. First, I summarize a qualitative interview study with 27 practitioners investigating how they understand and improve behaviors of complex AI systems. Next, I describe a theoretical framework that defines this process as a form of sensemaking and show how the framework can be used to create AI evaluation tools. I further show how insights into model behavior can improve human-AI collaboration by calibrating end-users' reliance on model outputs.

In the second half of the thesis, I implement two systems that, combined, fulfill the requirements of the full sensemaking process and BDAI workflow. I first introduce Zeno, an interactive platform that lets practitioners discover and validate behaviors across any AI system. I then describe Zeno Reports, a no-code tool built on Zeno for authoring interactive evaluation reports. Through case studies and real-world deployment with more than 500 users, I show how AI analysis tools covering the sensemaking process can empower practitioners to develop more performant and equitable AI systems.

ACKNOWLEDGEMENTS

This dissertation was only possible because of the advisors, mentors, friends, and family members who were there for me during the past five years, pushing me to excel during the peaks and picking me up from the troughs. A PhD is, more than anything, a journey of self-discovery. It forces you into an introspective deep dive, developing research taste and the ability to ask and answer the right questions while refining your values, goals, and the type of impact you wish to have. The people you surround yourself with define what this process looks like, and I will be forever grateful to those who helped me through it.

My advisors Adam Perer and Jason Hong were the cornerstone of my academic journey. Whether it was consoling me through the first few paper rejections or reining in my wild goose chases, they are the central reason I was able to cross the finish line. I will forever quote Jason's pithy sayings and stories and channel Adam's creativity and levelheadedness.

In addition to my advisors, I had three fantastic mentors on my committee. Since the first weeks of my PhD, I had the pleasure of meeting often with Ken Holstein. A uniquely deep and multidisciplinary thinker, I always came away from our conversations with a new framing for my work. I also had the fantastic opportunity to work with Ameet Talwalkar since the first year of my PhD, which kept me grounded in the world of machine learning and provided me with enlightening conversations. Lastly, I am deeply grateful to Aditya Parameswaran for being the essential external member of my committee and bringing unique questions and insights that helped shape the final form of this dissertation.

Beyond Carnegie Mellon, I had the opportunity to intern twice during my PhD with some incredible people. At Microsoft Research, Steven Drucker and Marco Tulio Ribeiro guided me through a seminal part of my thesis, starting with the simple question "How do you compare two models?". At Apple, I had the opportunity to work with Alex Bäuerle, Dominik Moritz, and Fred Hohman, three people that I am grateful to have had as colleagues and mentors beyond Apple. Fred was my mentor during undergrad, Dominik was part of the DIG lab at CMU, and Alex was my closest collaborator over the past few years.

I also grew substantially by mentoring other students. During my PhD I was fortunate enough to mentor over a dozen students, including Abraham Druck, Emily Guo, Kan Sun, Donny Bertucci, Tianqi Wu, Steven Huang, Erica Fu, Josh Zhou.

Inevitably, you develop deep bonds with your colleagues in a five-year journey. My labmates in the DIG lab, Will Epperson, Venkat Siviraman, Katelyn Morrison, and Frank Elavsky, provided constant feedback, laughter, and camaraderie. Beyond our lab, I also formed some of my deepest friendships, including Daniel Deroux, Mateo Dulce, Karan Ahuja, Rushil Khurana, Wesley Deng, and countless others.

Lastly, my family was the anchor that kept me grounded for the past five years. My parents, Elizabeth and Ángel, originally inspired me to pursue research and consider a PhD, and provided me with valuable perspective throughout the journey. My sister, Emilia, doubled as my roommate for an amazing year in Pittsburgh and always helps me take things a little less seriously. And Kristen, the love of my life - we rekindled our love, got engaged, and moved across the country during the PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Artificial intelligence (AI) systems have seen a surge of real-world usage in the past decade, with applications ranging from self-driving cars [1] and code-writing assistants [2] to cancer detection models [3]. The increase in usage can be attributed to two distinct waves of progress in AI research. First, in the mid-2010s, we had what is often dubbed the "deep learning revolution," the rise of neural network-based models such as convolutional and recurrent neural networks trained on labeled datasets. Researchers found that with a large enough model and sufficient labeled data, models could reach or surpass human-level performance in tasks previously thought impossible for machines.

So far in the early 2020s, we have seen an acceleration of the scaling trend in both the size of models and the training datasets. New hardware and model architectures allowed models to go from millions of parameters (ResNet [4] - 50 million, AlexNet [5] - 60 million) to billions (GPT-3 [6] - 175 billion). Self-supervision, training models on data without human labels, made it possible to train these new models on much larger datasets. Importantly, these models are not trained to learn specific tasks, such as image classification, but to learn general patterns in complex, unstructured data like language and vision. This makes it possible to use natural language prompts to create sophisticated AIs, greatly lowering the barrier to creating AI products and massively increasing their real-world deployment.

The growth in real-world use of AI has inevitably led to more people interacting with AI systems. This includes developers creating the AI products and end-users collaborating with the AI in their day-to-day lives. When a person interacts with an AI system, they develop a *mental model* of how the AI system works [7], as they do when interacting with any complex artifact [8]. The person uses their mental model to modulate their interaction with the AI system. For example, a developer may use insights into common model limitations to collect more data or update their text prompt. Or, a radiologist may use their mental model to decide to override the AI's prediction on certain X-ray images the AI tends to misidentify.

Unfortunately, forming accurate and robust mental models of AI systems is challenging, especially for modern AI systems outputting unstructured data like text or images. Although people develop mental models informally while interacting with an AI system, the traditional way to quantify the general performance of a model is to calculate an aggregate metric such as accuracy on a hold-out test set. This overview of AI performance is often insufficient, as it does not capture specific behaviors important for AI development and use, such as systematic failures, biases, safety issues, etc. People who work with AI need mental models that describe the specific *behaviors* of the AI, or what it tends to output for certain types of inputs.

In this thesis, I define a new AI development and deployment philosophy called **behavior-driven AI development** (BDAI) that places *behaviors* at the center of the process. BDAI makes behaviors the central unit of analysis for AI development and focuses on defining, quantifying, and communicating AI behaviors to improve model performance and deployment. In the BDAI framework, models are updated to change specific behaviors, and techniques to improve human-AI collaboration are aimed at improving people's mental models of AI behavior. Making behaviors central to AI development can lead to more performant and responsible AI systems better aligned with how we want AI to behave in practice.

The work presented in this thesis develops the BDAI philosophy with empirical studies and related frameworks and culminates in a general-purpose tool for behavioral analysis. First, I describe interview studies exploring how AI developers evaluate and iterate on their models in practice, finding that *behaviors* are the unifying abstraction across tasks and data types. Next, I formally define model behaviors and use a sensemaking lens to describe how developers understand and update AI systems. I further show that insights into behavior can improve human-AI collaboration by calibrating end-users' reliance. Lastly, I introduce two platforms, Zeno and Zeno Reports, that, together, fulfill each step of the sensemaking process for any AI system.

BDAI is an overarching philosophy that was introduced but not exhaustively explored in this thesis. In particular, the work described in this thesis focuses on how developers understand and validate behaviors and does not introduce specific tools to define or directly fix behaviors automatically. By defining the core abstractions and initial interfaces for BDAI, this thesis aims to inspire a growing body of work that continues to improve the development of responsible AI systems.

1.1 Overview

This dissertation is organized as follows:

In Chapter 2, I detail relevant background and related work. The first section briefly reviews the development of AI and how it relates to behavioral evaluation. I then provide background on sensemaking and how it has been a useful framework in related fields such as data science. Next, I touch on techniques for improving human-AI collaboration and show how sensemaking is important for nondevelopers. Finally, I survey a sample of important tools for different stages of AI development, such as error discovery, documentation, and reporting, and cover literature on narrative visualization.

Chapter 3 describes two interview studies conducted with AI practitioners. The first interview study, with 18 AI developers in 18 unique institutions, explores the challenges of AI development and the limitations of existing tools. The second study, with nine participants at Apple, dives deeper into practitioners' use of interfaces for AI development. We found that AI development is complex and multidimensional and that many traditional evaluation tools are insufficient for describing modern AI systems' behaviors.

Chapter 4 details a framework based on sensemaking that describes how developers make sense of AI behavior and formalizes the findings from the interview studies. It includes a survey of interviews with practitioners and AI tools used to adapt sensemaking specifically to AI development. I then describe the resulting framework and the AIFinnity system we developed to validate the framework.

In Chapter 5, I introduce *behavior descriptions*, details of model performance on subgroups of data. We show that behavior descriptions can improve human-AI collaboration in three distinct tasks. This study highlights how the abstraction of behaviors can be useful for non-developers interacting with AI systems.

Chapter 6 describes the Zeno system, an interactive platform for behavior-driven AI built using the sensemaking framework from Chapter 4. Zeno is a general-purpose evaluation tool that enables practitioners to perform fine-grained evaluation of any AI system. It consists of an extensible Python API for defining evaluation building blocks and an interactive user interface that practitioners use to discover and validate model behaviors interactively. We show how practitioners discovered significant model issues using Zeno in four case studies of tasks with different data and models.

Lastly, in Chapter 7, I introduce Zeno Reports. Building on the Zeno platform, Zeno Reports allows users to create more complex charts and instance visualizations and combine them into interactive and shareable reports. We publicly deployed Zeno and Zeno reports to over 500 users and explored how novice and expert users could use Zeno Reports to communicate complex findings.

1.2 Thesis Statements

Behavior-driven AI development is predicated on some core hypotheses defined and tested in this thesis.

First, I hypothesize that AI *behaviors*, metrics calculated on subgroups of data, are a useful abstraction to describe and update mental models of any AI system, regardless of the type of data or task. A robust mental model of AI behaviors is essential for building or working with an AI system.

Next, I hypothesize that developing mental models of AI behaviors is an iterative *sensemaking process*. Sensemaking is a useful high-level framework that has guided tool development in related fields such as data analysis, and I hypothesize that it can similarly help formalize AI analysis.

Lastly, I hypothesize that an AI-specific sensemaking framework can be used to create powerful analysis tools to discover and quantify AI behaviors. Using the specific stages and requirements from the sensemaking framework can lead to more thorough AI tools that cover all aspects of behavioral analysis.

1.3 Research Contributions

The work in this thesis presents several novel studies and systems. The initial interview studies make up the first empirical study directly exploring how AI developers evaluate complex AI systems and identifying gaps in tooling for effective AI development. The subsequent sensemaking framework is a novel formalization of how developers define and test AI capabilities. Behavior descriptions are a novel intervention for human-AI collaboration that improves end-users' calibration on AI outputs. Lastly, the Zeno system and Zeno Reports are the first general-purpose platforms for behavioral analysis that work across any AI task or data type.

1.4 Prior Publications and Authorship

This thesis results from work in collaboration with my advisors, Adam Perer and Jason I. Hong, and collaborators at Carnegie Mellon, Microsoft Research, and Apple AI/ML. The research which has been previously published is the following:

> Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. "Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning." *In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 419, 1–14.*
>
> Alex Bäuerle[1], Ángel Alexander Cabrera[1], Fred Hohman, Megan Maher, David

[1]Denotes equal contribution.

Koski, Xavier Suau, Titus Barik, and Dominik Moritz. 2022. “Symphony: Composing Interactive Interfaces for Machine Learning.” *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI ’22). Association for Computing Machinery, New York, NY, USA, Article 210, 1–14.*

Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M. Drucker. 2023. “What Did My AI Learn? How Data Scientists Make Sense of Model Behavior.” *ACM Trans. Comput.-Hum. Interact. 30, 1, Article 1 (February 2023), 27 pages.*

Ángel Alexander Cabrera, Adam Perer, and Jason I. Hong. 2023. “Improving Human-AI Collaboration With Descriptions of AI Behavior”. *Proc. ACM Hum.-Comput. Interact. 7, CSCW1, Article 136 (April 2023), 21 pages.*

CHAPTER 2
BACKGROUND & RELATED WORK

To contextualize BDAI, I provide an overview of related work in AI development, sensemaking, human-AI collaboration, and reporting.

2.1 What is AI development?

AI development is designing and implementing learned software that can perform complex tasks that typically require human expertise. The methods used to implement AI have changed dramatically over time, from rule-based expert systems to deep learning models [9, 10] and the recent wave of large foundation models [11]. Modern AI development has converged on data-driven models that can learn complex patterns from training data. Developers define the requirements of an AI system, collect representative data on the task or domain, and train stochastic models with the data [12].

The requirements of AI systems have historically been limited to high-level task definitions, such as classification or regression, measured with aggregate metrics on held-out validation and test datasets. Despite the proliferation of this approach, a large body of work over the past decade has found that this overly simplistic definition can lead to inadequate and potentially harmful systems. For example, an evaluation of facial recognition systems, Gender Shades, found that services deployed by major enterprises performed significantly worse for people with darker skin than those with lighter skin [13]. Since the Gender Shades work, numerous studies and examples of biases and safety issues in AI systems have emerged, from disparities in medical screening models to self-driving cars.

To detect and mitigate these important issues, the ML community has focused on the specific *behaviors* of AI systems [14, 15] beyond aggregate metrics. Inspired by requirements engineering in software engineering, behavioral evaluation focuses on defining and testing the capabilities of an ML system and its expected behavior in a specification of requirements [16, 17]. For example, a practitioner creating a sentiment classification model might check that the model works for double negatives, is invariant to gender, and is accurate for short text. In addition to aggregate metrics, they would check how their model performs in these specific scenarios.

A central challenge in behavioral evaluation is to decide *which* capabilities a model should have. There can be practically an infinite number of requirements in complex domains that would be impossible to list and test. Instead, ML engineers work with domain experts and designers to define the capabilities of a model as they iterate on and deploy their ML systems [18]. As end users interact with the model in products and services, they also provide feedback on the limitations or expected behaviors used to update the model [19].

In this thesis, I conduct a series of interviews with AI developers that confirm that behaviors are a core abstraction that can be used to define and improve model performance. These interviews add to the literature on empirical studies of AI developers by exploring in detail how developers discover, define, validate, and fix AI behaviors. Future work can explore how developers' processes change or what new challenges arise when they use tools built specifically for BDAI.

2.2 Sensemaking

If model development is about understanding and improving model behavior, what *are* behaviors, and what does it mean to understand them? In Section 4 we frame behavioral analysis as a sensemaking process. Sensemaking was initially formalized by Karl Weick, a social psychologist, in 1995 to describe how members of organizations come to a collective understanding of their surroundings [20]. At its most abstract, it can be defined as "structuring the unknown," or the "process through which individuals work to understand new, unexpected or confusing events" [21, 22]. It is an ongoing and iterative process by which people develop mental models of the world to make decisions and take actions. Weick's formalization of sensemaking spurred numerous empirical and theoretical studies, ranging from how organizations work through crises [23] to how entrepreneurs deal with failure [24] and even how we should design explainable AI [25].

Sensemaking has since expanded beyond social psychology and has been applied to domains such as ecology [26] and medicine [27]. Most relevant to this work are the applications of sensemaking to human-computer interaction (HCI), where computer and information scientists framed data analysis as sensemaking, constructing a mental model from extensive unstructured data. One of the first formalizations came from Russell *et al.* [28], who defined a "learning-loop complex" in which analysts cycle between creating representations of a system and fitting data to those representations. Russell's framework was later expanded by Pirolli and Card [29] to describe the specific steps and representations they observed data analysts use in practice.

Pirolli and Card [29]'s framework has become a frequent reference for data analysis and visualization research. One application of the framework has been structuring empirical studies of analysts, such as Grigoreanu *et al.* [30]'s study of programmers' processes and challenges when debugging software. It has also been used to design data analysis tools, including visualizations for large graph networks [31] or tracking patterns in microblogs such as Twitter [32]. Researchers and developers have been able to create tools that fit people's processes better by using Pirolli and Card [29]'s sensemaking framework.

In this thesis, I continue the rich history of sensemaking by adapting it to AI development using abductive analysis. Behaviors are the central unit of BDAI, and I use a sensemaking lens to formally define what AI behavior is and how developers understand and validate them in section 4.3.

2.3 Human-AI Collaboration

Developers are not the only ones who need to make sense of AI systems. Anyone who interacts with an AI system develops a *mental model* of how it behaves, when it performs well, when it fails, or when it has unusual results [33]. Mental models let people effectively work with AIs by helping them decide whether to rely on, modify, or override an AI's output. Therefore, it is important for people who collaborate with AI systems to have adequate mental models of AI to appropriately rely on the output of an AI [34]. There are various methods to encourage *appropriate reliance* of AI systems, often called *trust calibration* [35] techniques.

Studies have explored what factors influence people's mental models of AI systems. Kulesza *et al.* [36] found that people with more complete mental models could collaborate more effectively with a recommendation system. Bansal *et al.* [7] focused on the attributes of AI systems and found that systems with *parsimonious* (simple) and *non-stochastic* (predictable) error boundaries were the easiest for humans to work with. Other factors such as stated and perceived accuracy [37], confidence values [38], and model personas [39] can also influence people's mental models and their reliance on AI.

Recent methods have explored improving people's mental models, including tutorials explaining a task [40], tuning a model to better match human expectations [41], or adaptive trust calibration [42]. Some methods for improving mental models, such as adaptive trust calibration, use model details such as calibrated confidence scores to improve reliance. Another related method is Mozannar *et al.* [43]'s exemplar-based teaching of model behavior learns the nearest neighborhood regions around model failures to help validate people's mental models.

Existing methods attempt to tune end-users mental models of AI systems using proxy methods such as tutorials or forcing functions. Inspired by the idea of behaviors being the core unit of model performance, this thesis shows that telling end-users directly how a model behaves for subsets of data can calibrate human reliance on AI aids. Given the initial promise of these behavior descriptions, future work can explore what are the most effective methods of creating and showing these insights.

2.4 Tools for AI Evaluation

AI behavior is often complex and multidimensional, and human-centered tools can help developers discover, validate, and track model behaviors. The most direct tools for behavioral evaluation directly define and measure behaviors. One such example is Checklist, a system that uses a mix of evaluation methods such as minimum functionality and metamorphic tests to uncover fundamental limitations of language models [44].

Model behaviors are numerous and often unknown *a priori*. Interactive tools can enable users to use their domain knowledge and expertise to discover and validate errors. ModelTracker [45] was one of the first interactive systems that helped developers go beyond aggregate accuracy and better understand the specific behaviors of classification models using a unit visualization. Since then, many tools have been developed for specific models and behaviors. Errudite [46] is a system for interactively debugging language models using templating filters and rewriting rules. FairVis is another example of an interactive system specific to intersectional fairness issues [47].

There are also many algorithmic methods to discover model limitations automatically. If a dataset has generous amounts of tabular data, SliceFinder [48] is an efficient method for discovering intersectional subgroups of data. For more complex data types without clear tabular data, blindspot discovery methods find rough subgroups of data with high error, such as Domino [49]. With the growing use of prompt-based models that do not require labeled data, researchers are exploring methods for generating specific evaluation data. AdaTest is a system that uses a larger language model to generate evaluation instances to test smaller, task-specific models [50].

Zeno, the system for BDAI introduced in section 6.3, empowers users to find significant model failures across tasks and data types. It takes inspiration from existing methods for behavioral evaluation, like Checklist, but provides a more general interface for defining errors that are not formally pre-defined. Intelligent features such as SliceFinder or Domino can be added to Zeno to make it easier to discover errors that can still be represented and tracked using the core behavioral abstractions.

2.5 AI Documentation and Reporting

Various documentation methods help practitioners track and communicate details about their models. Model Cards [51] and FactSheets [52] include important information and details about machine learning models. These model reports include information ranging from the model type and hyperparameters to aggregate metrics and ethical considerations.

Most ML models are developed by cross-functional teams with stakeholders in technical and non-technical roles. While collaboration is essential to decide how a model should behave and identify potential failures, there is often limited communication between stakeholders [53]. This can lead to unrealistic expectations of model performance or results that do not match designers' expectations. Multiple methods have been proposed to improve organizations' shared understanding of model behavior.

Interactive systems have shown promise in bridging model knowledge between engineering and other roles. An example framework, Symphony [54], which I worked on at Apple, introduces modular data and model analysis components that can be used in computational notebooks and standalone dashboards to allow more stakeholders to explore model behavior. Marcelle [55] uses modular components that allow users to modify an ML pipeline without writing code.

Complex models also require robust reporting methods to ensure that information about data and models is recorded and preserved. Datasheets for Datasets [56], FactSheets [52], Nutritional Labels [57], and Model Cards [51] codified the first principles for documenting ML details for future use and reproducibility. Extensions to these reporting methods, namely Interactive Model Cards [58], have aimed to improve their usability by making them more expressive and interactive.

Zeno Reports, presented in this thesis, combines the approaches of the opinionated reporting specifications, like Model Cards, with the live data approach in more complex systems such as Symphony. Zeno reports are directly tied to the underlying data, but are limited to core visualizations highlighting behavior.

2.6 Narrative Visualization

To design Zeno Reports, we drew inspiration from the concept of *narrative visualization*, a term popularized by Segel and Heer [59]. Narrative visualization combines storytelling techniques with (often interactive) visualizations, enriching the narrative with data-driven insights. This approach is characterized by its ability to communicate complex data in an engaging and easily digestible manner, its emphasis on guiding the audience through a data-driven story, and its interactive elements that invite audience participation.

Notable systems developed for this type of storytelling include Idyll [60] and Idyll Studio [61]. Idyll facilitates the creation of interactive articles that combine narrative text with data-driven content, while Idyll Studio extends this functionality with a more intuitive, user-friendly interface. In addition to academic systems, many commercial visualization systems, such as Tableau and Quarto, support creating linear narrative stories around visualizations using their native interfaces.

These systems are designed to support the broad spectrum of visual storytelling and thus are complex or rely heavily on coding skills or specific markup languages. In contrast, Zeno Reports offers a simpler, no-code interface focusing exclusively on AI evaluation. This specialization allows us to streamline the user experience, making it more accessible to users who may not have advanced programming skills yet are still powerful enough to convey the complexities of AI model performance effectively.

CHAPTER 3
HOW DO PRACTITIONERS EVALUATE AI SYSTEMS?

This first section of the thesis describes two interview studies conducted as need finding studies for two implemented systems, *Symphony* [54] and *Zeno* [62]. It explores the core practices and challenges in modern AI development, specifically evaluation workflows and analysis interfaces, and sets the stage for developing a behavior-based view of AI development.

This chapter was adapted from the following published papers:

> Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. "Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning." *In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 419, 1–14.*
>
> Alex Bäuerle[1], Ángel Alexander Cabrera[1], Fred Hohman, Megan Maher, David Koski, Xavier Suau, Titus Barik, and Dominik Moritz. 2022. "Symphony: Composing Interactive Interfaces for Machine Learning." *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 210, 1–14.*

3.1 Introduction

As the architectures and algorithms for training AI systems have evolved, so have the processes and tools for creating AI products. This includes everything from data labeling and training workflows to evaluation and debugging methods. We wanted to understand AI developers' challenges when creating and improving their AI systems, especially practitioners working on more complex models, such as deep learning models on non-tabular data. We also wanted to dig deeper into what tools they currently use and what gaps exist in the tooling landscape.

[1]Denotes equal contribution.

This section describes two interview studies we conducted to answer these questions. The first set of interviews was with practitioners in small and medium-sized companies focused on their challenges around the evaluation and iteration of AI systems. The second is a series of interviews conducted at Apple with nine practitioners in seven teams working on AI products. These interviews focused specifically on the tooling these teams used or wished for in AI development.

The concrete research questions that we aimed to answer in these interviews were the following:

RQ1. How do AI developers evaluate and fix their models over time?

RQ2. What tools and approaches are developers using to evaluate and fix their models?

RQ3. What are the limitations of existing tools?

Both studies found similar trends and patterns in the processes and tools of AI developers. Overall, we found that current AI evaluation and iteration tools are insufficient for the complexity of modern AI systems. Specifically, they do not support the disaggregated analyses required to identify important behaviors and create robust, deployable systems.

3.2 Methodology

3.2.1 Interviews on AI Evaluation

We conducted semi-structured interviews with machine learning practitioners to explore our first research question: What are the common challenges for ML evaluation in practice? In particular, our objective was to understand the specific challenges that practitioners face and the tools they use when evaluating ML models. The 18 participants, listed in table 3.1, hold various roles related to machine learning development and deployment, from data scientists to CTOs and CEOs of small companies. The initial participants were recruited through posts on social media networks, e.g., Reddit, LinkedIn, and Discord, as well as direct contacts at technology companies. Additional participants were then recruited through snowball sampling. Each interview was conducted via video call and was, at most, an hour long. The study was approved by our Institutional Review Board (IRB) and the participants were compensated with $20.

Two researchers analyzed the interviews using inductive iterative thematic analysis and affinity diagramming. The researchers extracted common themes around model evaluation, debugging, and iteration from the first few interviews, grouping similar findings in an affinity diagram. After each subsequent interview, the researchers iterated and refined the themes as needed. The recruitment of new participants was stopped when no new themes were produced from the last few interviews.

3.2.2 Interviews on AI Interfaces

To understand how ML interfaces are used in practice, we conducted seven semi-structured interviews with 9 participants at Apple. We recruited participants through internal emails and messaging boards. We selected participants in various roles, including engineers, designers, researchers, and testing roles that work in teams to build and deploy ML systems. Each interview was conducted through a video call and lasted about an hour. First, we asked participants how they create and use different ML interfaces such as documentation, visualization dashboards, and widgets. We then asked them about the main limitations and pain points in current tools and what types of improvements they would find helpful.

3.3 Results

3.3.1 Aggregate Metrics Do Not Reflect Model Performance in Deployment

All practitioners (18/18) focus on improving aggregate metrics when developing new ML models, but, as P9 admitted, you *"can perform very well on a training dataset, but when you go to ship the product, it doesn't work nearly as well."* To ensure that models perform as expected when deployed, all practitioners also evaluate their models in real-world use cases. For example, P16 evaluates their text analysis model on a per-client basis since they had found that their model underperformed for certain types of data, e.g., healthcare notes, that it was not trained on. This type of behavioral analysis is often also called *qualitative* analysis, looking at specific instances and model outputs to confirm hypotheses of model behavior.

There are various methods practitioners described for discovering model limitations and failures, from end-user reports (see Section 3.3.3) to automated clustering algorithms. A common technique 11 of the 18 participants mentioned was creating data inputs to probe a model and find potential failures, often called "dogfooding" in software development. For example, when selecting an audio transcription service P3 *"has some data collected we recorded ourselves, and then we pass it to different services and explore the structure of the output"* to decide which service provides the qualitatively "best" output for their task. Two participants are exploring automated error discovery methods such as finding clusters with high error or using foundational models [11, 50] to generate new instances, but still rely mainly on human-generated feedback.

After generating hypotheses of systemic failures, many practitioners craft test sets to validate how prevalent behaviors are (10/18). The participants had different terms for these instances, including "golden test sets", "dynamic benchmarks", "regression tests", and "benchmark integration tests". Despite the varied terminology, these tests have the same structure: Expectations of model outputs on different subgroups of instances. For example, P4 has multiple text inputs with common human typos and valid outputs that they check before models are released.

None of the participants who conduct this type of behavioral evaluation use standardized frameworks. This is primarily because existing behavioral evaluation tools do not work for their data or model types, so they develop their own tools, such as scripts or web interfaces, to monitor model performance. All the participants who do not perform behavioral analyses (8/18) wish to conduct more detailed testing, for example, P1 wants *"to do some other testing, but we don't do anything because there's not a really easy to set up system to do that"*. In general, larger companies can dedicate more time to a detailed evaluation and the building of customized tools that smaller companies cannot afford despite their need for more comprehensive evaluation [63].

While all participants detailed clear use cases for ML interfaces, they also mentioned limitations preventing them from using existing tools or sharing insights. One participant bluntly stated *"right now, we basically have no tools"* for analyzing ML systems. Instead, participants rely on ad-hoc, hand-crafted visualizations for their specific analyses. For example, one of our participants said that their process to look at instances is to *"manually examine icons in a file explorer."* Another participant *"looks at handcrafted summaries of select data subsets"* to do model analysis. Larger teams with more resources may have customized tools, such as a participant that *"use[s] a team-internal tool to analyze data"*. Overall, inadequate tooling leads developers to use one-off, manual tools or ML teams investing in custom visualization systems.

3.3.2 Challenges in Tracking Continuous Model and Data Updates

All practitioners (18/18) we interviewed update their models as they design better architectures, gather more data and discover real-world use cases and failures. Participants described this process with different terms, such as "rapid prototyping" or "agile" methods, in which they quickly act on user feedback and deploy updated models. P4 and P13 even started with "wizard-of-oz" models with a human emulating an AI or non-ML models to gather data and model requirements before developing more complex models.

Although updating a model can improve the overall performance of an ML system, it can also lead to new failures. This is especially true for stochastic models, such as deep learning, which cannot be deterministically updated. As P5 lamented, *"our test set would become so large that if we had to fail for less than 5 [tests] it became super hard to make progress"*. Model updates are even more complicated for teams that rely on external AI services, as practitioners do not control when or how services are updated [64]. For example, P3's team had to switch their voice-to-text service from Google to Amazon because Google stopped detecting filler words such as 'um' after a model update, which was necessary for their product.

Due to these frequent updates, comparing models across important behaviors becomes important. However, since many model evaluations are run inconsistently and across different tools, the history of past performance is often fragmented or lost, making it difficult to find regressions or new failures.

3.3.3 Limited Collaboration in Cross-Functional Teams

Modern machine learning development in practice is a collaborative effort that spans different teams and roles. Each team member needs a robust mental model of how an ML system behaves to resolve customer complaints, make management decisions, validate failures, and more.

A common collaboration challenge is making sense of failure reports [19] from end-users. 12 of the 18 participants' teams have customer service representatives who parse tickets or complaints from end users and pass them to the engineering teams. These participants found it challenging to reproduce the reports from the end users, which were primarily made up of one-off instances and broad descriptions. P4's team tackles this challenge with an *"internal website where anybody can put potential inputs and expected model outputs"* which new models are tested.

Another collaboration challenge described by 14 participants is communicating the model performance with managers and other stakeholders. For example, P16's management team often makes decisions based solely on a high F1 score, while it is often the case that different clients require different trade-offs between precision and recall. Many decisions about whether or not to deploy an updated model require shared knowledge and conversations between engineers, managers, and customers on whether a new model is holistically better than the existing model.

Since engineers often run analyses in ad-hoc scripts or notebooks, knowledge of model behavior can be isolated. Other stakeholders do not know how a model tends to behave, and can neither make informed decisions on model usage nor provide information about model errors to engineers for debugging.

Due to limited, isolated interfaces, participants described various challenges in communicating and sharing insights. Since different stakeholders prefer different environments, such as code-based notebooks or standalone dashboards, it can be challenging to share insights with others. In addition to sharable interfaces, participants also wanted cross-platform support for themselves. As one participant put it, *"I would like both an environment for experimentation and always there reliable visualizations."*

It can also be difficult to transfer visualizations and findings between platforms that different stakeholders work with. One participant lamented that *"I am often not invited to the table until things go wrong,"* and in some teams *"designers often times don't have access to data and model results."* In turn, decisions about ML systems are made without all team members having a shared understanding of the current state and limitations of the project. Despite these current limitations, the participants thought that *"fostering a culture of sharing insights would be great."*

3.3.4 Use Cases and Limitations of Development Tools

All participants agreed that creating and sharing ML interfaces can help them build more robust and capable ML products. Participants described use cases of ML interfaces in myriad tasks, such as *"flagging failures for review," "detecting systematic failures,"* and *"fairness and bias education."* Participants also mentioned stages throughout the entire ML process in which ML interfaces can be useful, from *"dataset curation and sharing"* to analysis *"after an ML model has been trained,"* or *"in all stages"*. Consequently, since different stakeholders involved in an ML product need specific views of the data and models, ML interfaces must be flexible enough to support analysis across numerous tasks and domains.

Participants detailed a variety of technical roadblocks and time-consuming processes that prevented them from using existing ML interfaces. Many tools require users to wrangle and export their data into a specific format before loading them into a custom system or dashboard. However, as a participant stated, *"we do not have a lot of time for creating such visualizations:"* developers simply do not have the bandwidth to perform the setup and data wrangling work necessary to use separate systems. developers' main priority is working on data and models, and *"if it takes longer than 5-10 minutes, I am not going to [use an ML interface] immediately"*.

Five participants explicitly mentioned that they do not use ML interfaces because they are not available in the environments where they work and that *"people would want to use easier tools."* For example, *"many data scientists want to explore their data in notebooks"* without having to open a separate system. Additionally, since data and models update frequently, one participant wanted to *"start a job with checkboxes and buttons"* and produce a self-updating web UI that they would not have to manually author.

Lastly, the teams we talked to work with myriad data types, such as video, 3D point cloud, tabular, image, and audio data, and desired bespoke visualizations supporting their analysis needs. One participant mentioned running and visualizing specific data analyses and *"would want to specify algorithms because our problems are very specialized."* However, current data science tools often only provide visualizations for a limited set of data types and models.

3.4 Conclusion

Across these interviews, we found that developers of AI systems focus their efforts on identifying and improving specific model behaviors that users see in production. We found that existing tools for evaluating and iterating on models mainly focus on academic benchmarks and traditional domains such as classification, which do not reflect the complexity of commonly deployed models and user inputs. Tools that can track behaviors across data and model updates and that can be used by diverse stakeholders could better support the development of responsible AI systems.

Table 3.1: The practitioners in the semi-structured AI evaluation interviews (Section 3.2.1

ID	Role	Area
P1	AI Software Engineer	AI Consulting
P2	Data Scientist	Clothing Retail
P3	CTO	Speech Training
P4	CTO	Voice Assistant
P5	Senior ML Engineer	Chatbot
P6	Data Scientist	AI Non-profit
P7	Data Scientist	Finance
P8	MS Student	Educational Technology
P9	ML Engineer	Chatbot
P10	VP of Data Science	Business Intelligence
P11	ML Engineer	AI Explainability
P12	Data Scientist, ML	Ridesharing
P13	Data Engineer	Educational Technology
P14	CTO	Health Technology
P15	CEO	Sensing
P16	Data Scientist	Search and Recommendation
P17	ML Research Scientist	Epidemiology
P18	Data Scientist	Video Streaming

CHAPTER 4
A SENSEMAKING FRAMEWORK FOR BEHAVIORAL EVALUATION

This chapter formalizes insights from the interview studies into a sensemaking-based framework describing how developers analyze AI behavior. It includes an extended review of existing empirical studies of developers, a prototype system that implements each step of the sensemaking process, and a think-aloud study. This framework is the backbone of the BDAI philosophy, as it explicitly defines AI behaviors and the canonical steps to discover, validate, and fix them.

This chapter was adapted from my published paper:

> Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M. Drucker. 2023. "What Did My AI Learn? How Data Scientists Make Sense of Model Behavior." *ACM Trans. Comput.-Hum. Interact. 30, 1, Article 1 (February 2023), 27 pages.*

4.1 Introduction

Designers make sense of feedback to inform their designs [65], doctors make sense of health records to guide their diagnoses [66], and programmers make sense of code to debug their software [30]. Similarly, data scientists make sense of their machine learning (ML) or artificial intelligence (AI) models to improve their performance, decide when to use them, and analyze their real-world impacts. Having a thorough understanding of how an AI behaves is especially important to detect and mitigate serious concerns such as fairness [67] and safety [68] issues.

What does it mean to make sense of AI behavior? Let us explore the example of a data scientist who wants to make a website more accessible by including text descriptions (alt-text) for images. They find multiple AI services for captioning images and have to pick the option that works best for their data. The data scientist compares the options by generating alt text with each AI for a sample of images and develops a mental model of how each AI *behaves*: which AI can describe certain activities, is better in low light, or is more grammatically accurate. With a deeper understanding of how each AI service behaves, the data scientist can decide which one to use for their data. This is just one example use case for understanding AI behavior, which is essential for tasks ranging from training new models to detecting dataset shift and mitigating real-world failures.

While important, behavioral analysis requires significant human attention to ideate, structure, and test hypotheses of AI behavior. Data scientists instead often resort to limited and ad hoc methods, such as manually testing edge cases or waiting for end-users to report failures of deployed models [67, 63, 12, 69]. A number of AI analysis tools aim to improve this process, including crowdsourcing methods for discovering failures [70, 19], algorithms for finding slices of data with high loss [48], and checklists of expected model behavior [44]. Although useful for specific tasks, these tools tend to only address portions of the analysis process and are hampered by challenges at other stages of the process. For example, methods for creating subgroups of data [48, 71, 47] do not tell the user *which* subgroups are the most important, while model checklists do not have mechanisms for discovering new behaviors.

This article introduces a sensemaking framework describing how data scientists develop mental models of AI behavior. By framing AI analysis as sensemaking, we aim to provide a language for describing AI analysis, help developers identify gaps in existing tooling, and encourage analysis tools supporting the full sensemaking process. Sensemaking is a well-established paradigm that describes how people structure the unknown by iteratively creating mental models from data [20]. To accurately describe AI analysis as sensemaking, we used abductive analysis to adapt Pirolli and Card [29]'s framework for data analysis to fit the steps specific to AI development gathered from empirical studies of practitioners. Our resulting framework (Figure 4.1) describes how people create mental models of AI behaviors by organizing instances into meaningful schemas and hypotheses. The *mental models* data scientists derive are their internal representations of the behaviors of a complex, often black-box, AI *model*.

We evaluated our framework across the three powers of interaction frameworks defined by Beaudouin-Lafon [72]: *descriptive*, *evaluative*, and *generative* power. To test the framework's *descriptive* and *evaluative* power, how it can detail and compare a range of existing interfaces, we reviewed AI analysis tools and showed how they fit into the stages of our framework. We found that most tools only address half of the sensemaking process, either discovery tools for finding and organizing instances or evaluation tools for testing known behaviors. Systems that combine discovery and evaluation could help data scientists effectively validate newly discovered behaviors. Next, to directly test our framework's

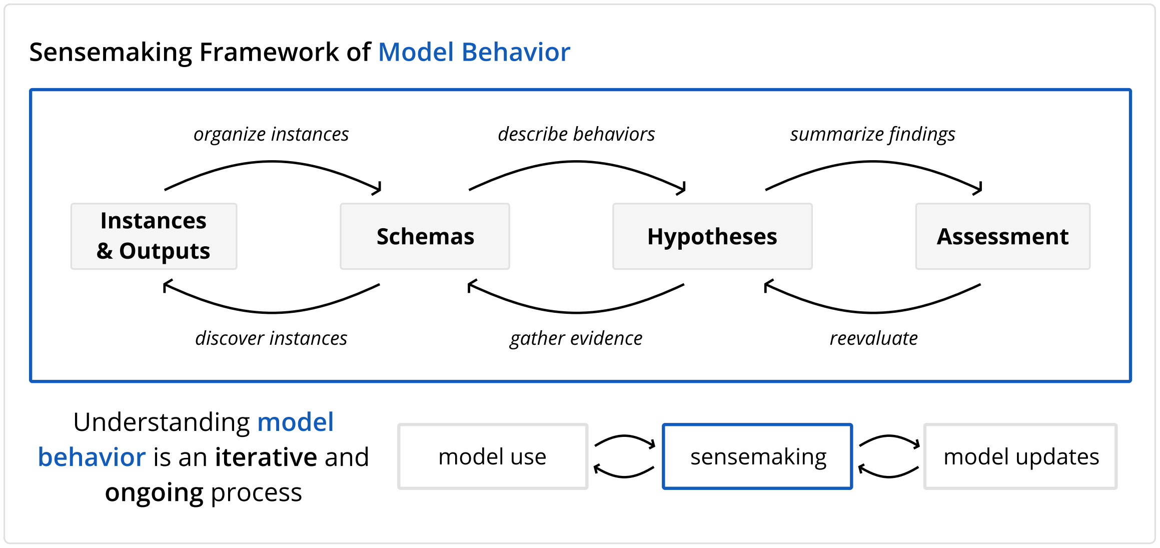


Figure 4.1: The sensemaking framework describing how data scientists understand model behavior. We derived the framework from [29]'s sensemaking process and empirical studies of data scientist. The process is iterative and ongoing, with data scientists continuously reevaluating as they update and deploy their models.

generative power, the ability to inform new designs, we used it to create an AI analysis tool, AIFINNITY, for exploring image-and-text models like visual question answering and image captioning. Image-and-text models have many complex behaviors, from stereotypes to grammar issues, that make them a challenging domain for AI analysis.

For our final evaluation of the framework, we explored how data scientists use a full sensemaking system. We conducted exploratory think-aloud studies with 10 professional data scientists tasked with using AIFINNITY to choose between two image captioning AIs. Participants found that AIFINNITY matched their mental process for understanding AI behavior, with some even independently describing their processes in sensemaking terms. Additionally, the complementary features helped participants find numerous significant behaviors and actively think about confirmation bias.

In summary, the main contributions of this work are the following.

- A **sensemaking framework** describing how people develop mental models of AI behavior.
- An **AI analysis tool** called AIFINNITY designed using the framework.
- An **exploratory think-aloud study** with 10 professional data scientists to understand how people work with an AI analysis tool for the full sensemaking process.

4.2 Methodology

To create a framework that describes AI practitioners' process we used *abductive analysis* [73] to iteratively adapt Pirolli and Card [29]'s sensemaking framework to empirical studies of AI/ML practitioners. In contrast to *inductive* methods such as grounded theory [74], which develop a framework from empirical evidence, and *deductive* approaches that directly apply existing theories, *abduction* extends or develops theory to explain new evidence. We decided that an abductive approach would be the most appropriate for this work since we adapt theory from a related domain, data analysis, to describe a new process, how practitioners understand AI behavior.

We primarily built from Pirolli and Card [29]'s sensemaking framework which describes how intelligence analysts make sense of large amounts of unstructured data. In their framing, analysts first go through an information foraging loop, where they filter *data sources* into a *shoebox* of relevant information. Snippets from documents in the shoebox make up the *evidence file*. Next is the core sensemaking loop, where analysts create *schemas*, structured organizations of the data, from the evidence file which are used to create and support *hypotheses*. Lastly, these hypotheses are used to create a final *presentation*. One can imagine a detective in front of a corkboard, cutting out and organizing newspaper clippings to pin them up and connect them with red thread.

To adapt Pirolli and Card [29]'s framework to AI analysis, we reviewed empirical studies of how practitioners work with AI systems in the real world. Since there are no survey papers, to date, directly covering this area, we relied primarily on academic search engines and citation graphs. Our review focused on studies with first-hand interviews and surveys to get the most direct look at data scientists' processes (Table 3.1). For our analysis, we coded the empirical studies and used an affinity diagram to recursively fit the codes to the [29] sensemaking stages. During the abductive analysis, we also updated the stages to better describe AI practitioners' processes. In the following section, we describe the resulting framework in detail and describe the key ways in which it differs from existing frameworks.

4.3 Sensemaking Framework

The resulting sensemaking framework for understanding AI behavior is shown in Figure 4.1. The least structured stage is gathering **(1) instances and model outputs** from a variety of sources such as real-world users or synthetic methods. Data scientists then begin to organize the instances into general **(2) schemas** of semantically similar instances and behaviors. Schemas can be either rough groupings or strict slices of data. Data scien-

Table 4.1: How existing AI analysis systems fit in the sensemaking framework. Some tools focus on specific behaviors, like biases, or domains, like self-driving cars, but they all help data scientists better understand the behaviors of their AI systems at different points in the sensemaking process.

Venue	Paper	Instances	Schemas	Hypotheses	Assessment
AAAI	Beat the Machine [70]	🟦			
arXiv	Dynabench [75]	🟦			
ICLR	[76]	🟦			
CVPR	StyleGAN [77]	🟦			
JBD	Data Augmentation [78]	🟦			
VIS	CAVA [79]	🟦			
VLDB	Snorkel [80]	🟦			
WWW	Patterned BTM [81]	🟦	🟦		
VIS	What-if Tool [82]	🟦	🟦		
HCOMP	Pandora [83]		🟦		
AAAI	Lakkaraju *et al.* [84]		🟦		
arXiv	Spotlight [85]		🟦		
CVPR	Barlow [86]		🟦		
ICDE	Slice Finder [48]		🟦		
VIS	FairVis [47]		🟦		
CHI	ModelTracker [45]		🟦		
VIS	Squares [87]		🟦		
N/A	Facets [88]		🟦		
IUI	AnchorVis [89]		🟦		
HILDA	MLCube [71]		🟦		
CSCW	Deblinder [19]	🟦	🟦	🟦	
ACL	Errudite [46]		🟦	🟦	
ICLR	Domino [49]		🟦	🟦	
VIS	HypoML [90]			🟦	
ASE	DeepRoad [91]			🟦	
ICSE	DeepTest [92]			🟦	
ICSE	Invariant Testing [93]			🟦	
FAccT	Interactive Model Cards [58]	🟦	🟦		🟦
CHI	Symphony [54]		🟦		🟦
arXiv	Robustness Gym [94]		🟦	🟦	🟦
ACL	Checklist [44]			🟦	🟦
FAccT	Model Cards [51]				🟦
IBM JRD	FactSheets [52]				🟦

tists then define formal **(3) hypotheses** of AI behaviors and gather additional evidence to validate their hypotheses. Lastly, data scientists derive a final **(4) assessment** of their discoveries, organizing hypotheses to be useful in subsequent tasks like choosing between AI services or updating a model's architecture. The sensemaking process does not have to start from the initial stage of instances and outputs. Practitioners may have existing hypotheses, or may use tools that slice and organize instances into pre-defined schemas.

This adapted framework differs in a few key ways from the Pirolli and Card [29] formalization. Primarily, it is missing the initial foraging loop with the *shoebox* and *evidence file* stages. Unlike analysts who sort through data sources, such as newspapers, to extract snippets of evidence, AI analysis starts with instances, model inputs, that are directly relevant to a model's behavior. While AI practitioners actively search for new instances to discover hypotheses, they do not have to further sort and modify instances in their sensemaking process. Next, the instances and outputs in AI analysis are lower level than the data sources, like research articles, used by analysts. Thus, the schemas for AI analysis tend to be groupings of instances rather than connections between high-level patterns or findings. This also means that hypotheses are directly verified using supporting instances and outputs, and need sufficient, diverse evidence to be accurately evaluated. Overall, the focus of AI analysis is on creating appropriate schemas and ensuring the validity of hypotheses rather than foraging for relevant evidence.

The context in which AI analysis occurs also differs significantly from sensemaking in domains such as data analysis. Sensemaking for AI systems is an iterative and ongoing process, as AI systems are constantly being updated and applied to new domains. In traditional data analysis, new reports or research may update existing hypotheses over time but often do not lead to brand new patterns. Updates to black-box AI systems, on the other hand, can completely change the behavior of an AI system and require reevaluating all hypotheses. Additionally, new instances are constantly being received from end users, informing new schemas and hypotheses. The volatility and quick iteration of AI systems have implications for tools that support the sensemaking process.

In the following sections, we describe in detail the four stages of the sensemaking process for AI analysis. In each section, we first describe how data scientists currently approach the sensemaking process and then describe existing tooling available at each stage.

4.3.1 Instances and Outputs

At the core of the sensemaking process are data instances and their associated model predictions, the outputs of the model for the given instances (Figure 4.2). The most convenient source of instances are datasets collected to train an AI system, often split into training,

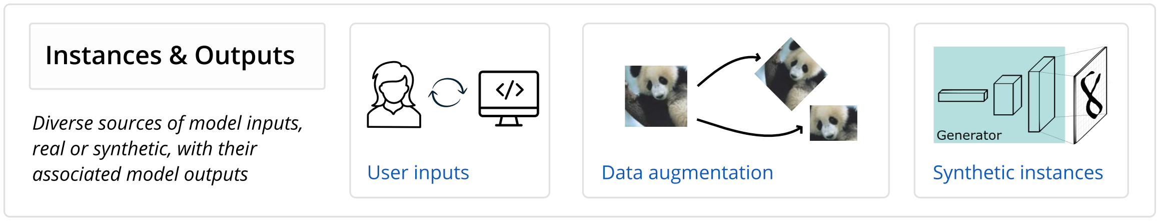

Instances & Outputs
Diverse sources of model inputs, real or synthetic, with their associated model outputs
User inputs
Data augmentation
Generator
Synthetic instances

Figure 4.2: The least structured stage of the sensemaking process consists of **instances and outputs**, model inputs from a variety of sources along with their associated model predictions. Instances can include both real-world user inputs and synthetic data.

validation, and testing sets on which aggregate metrics are calculated. While convenient, initial training datasets are limited and can lead to misleading performance measures and missed behaviors. For example, one participant in Wan *et al.* [95]'s study found significant overlap between their training and testing sets that produced an inflated model accuracy, while two participants interviewed by Hopkins and Booth [63] lamented that they needed a much greater diversity of instances than they had to accurately evaluate the performance of their model.

To better understand the behavior of their models, data scientists constantly collect new real-world instances to both update their models and discover new behaviors. This is especially important due to data drift, with 55% of the data scientists interviewed by Sambasivan *et al.* [96] describing factors such as new environmental factors and human patterns leading to model failures or unexpected outputs. The data scientists interviewed described monitoring the performance of the model over time on newly collected instances to identify performance drops or new regressions.

Despite the utility of real-world data, it is often expensive and slow to gather and label real instances, limiting developer access to data. Instead, data scientists "dogfood" their models, creating instances they think might be particularly difficult for an AI or show interesting behaviors [69]. Data scientists interviewed by Hopkins and Booth [63] found that this type of "prodding and probing" of models helped them better understand and work with black-box systems. Dogfood testing can be especially important for rare or sensitive behaviors which could have serious consequences in the real world [12].

Finally, it is not just the quantity and diversity of instances that is important for AI analysis, but what features are available for each instance. For example, to detect whether a model treats people of a certain demographic group inequitably, the data instances have to have a feature for that demographic information. Sensitive information, such as demographic details, is often not collected or present in a dataset and was one of the primary challenges for data scientists in discovering biases found by [67]. In sum, both the number of instances *and* number of features of a dataset are important for discovering relevant behaviors.

Data collection and labeling methods

Tools at the instance and output stage often focus on scaffolding data collection, artificially generating instances, and adding features to a dataset.

Instead of waiting to gather real-world data from users, some techniques proactively use crowdworkers to gather instances. Beat the Machine (BTM) [70] and DynaBench [75] directly ask end-users to explicitly find instances for which a model fails, collecting instances that may surface interesting behaviors. Subsequent methods such as Deblinder Cabrera *et al.* [19] and Patterned Beat the Machine [81] build on this process by asking users to provide more context for a failure and find instances relevant for later schemas and hypotheses.

Data is often expensive to collect, so *synthetic*, artificially generated instances can provide a useful alternative to real-world instances. A common method for creating synthetic data is data augmentation, creating new instances by modifying existing ones, e.g., rotating or cropping images [78]. To create new instances that are not in a dataset, techniques like generative adversarial networks (GANs) can be used to generate novel examples [76]. StyleGAN is one such technique that generates new images from high-level semantic descriptions [77]. Synthetic instances are a low-cost way to augment a dataset, but it is not possible to generate any arbitrary instance, and synthetic instances are often less diverse than examples found in the real world.

There are also methods for adding new features to a dataset, providing details for each instance that can surface new behaviors. A separate AI model or heuristic functions are a common way to extract new features from an instance, such as the noisy labeling functions in Snorkel [80]. A related system is CAVA, which uses a knowledge graph to extract new attributes for an instance, such as populations from country names [79]. Additional features, or metadata, are essential for the subsequent stage of grouping and organizing instances into schemas.

Gathering diverse instances remains a challenging problem, as traditional methods remain expensive and synthetic techniques are noisy and limited to certain data types. In the context of the full sensemaking process, tools at the instance and output stage are often not informed by findings from later stages, such as interesting schemas or new hypotheses. For example, validating hypotheses requires collecting specific instances, which is often not well supported by current data collection methods. Data collection or generation techniques that are more closely informed by the needs of schemas and hypotheses could better support data scientists' AI analysis process.

4.3.2 Schemas

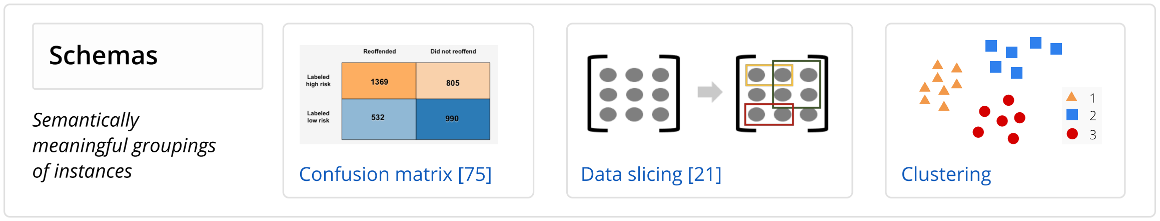


Figure 4.3: Creating **schemas** is the second major sensemaking stage. Schemas are organizations of instances into meaningful layouts or groupings. Common schemas for AI outputs include confusion matrices, subgroups of data, and clusters.

The second sensemaking stage is organizing instances into semantically meaningful groups, called *schemas* [29, 97]. Schemas let practitioners hypothesize new model behaviors or collect evidence for existing hypotheses (Figure 4.3). There is significant flexibility in how schemas are created, from formal slices of a dataset to rough groupings of semantically similar instances.

Some of the most common schemas are classic methods for evaluating AI systems, such as the confusion matrix for classification problems [98, 99] and residual plots for regressions. Yang *et al.* [100] described the use of these visualizations as core knowledge required by the data scientists they spoke with. Splitting a model's output by predicted and ground truth output lets data scientists identify numerous metrics related to the model's behaviors; does the AI have a higher recall than precision? Is the false positive rate acceptable? These questions of model behavior are often central for data scientists, such as data scientists in Wan *et al.* [95]'s study making tradeoffs between metrics like precision and recall. Residual plots give a similar idea of how well a regression model behaves, as nonrandom errors can suggest a model is not adequately describing the data.

While these output-based visualizations may be helpful, they are limited to detecting behaviors described by output groups. Many important behaviors are found in groups defined by a model's *input* features; for example, fairness issues are defined by demographic information that is rarely the output of a model. Often called 'subgroup analysis,' or 'data slicing,' splitting and comparing instances by input features can detect such behaviors. Data scientists often look at model performance across these subgroups to track issues such as biases [67, 63].

For less structured data types such as images it can be challenging to create groups of similar instances in the first place, such as all images with a specific object in them. Without additional metadata collected or generated in the instances stage, it is not possible to create clear schemas for those semantic features. To address this, a data scientist in Holstein *et al.*

[67]'s study wished for an oracle that would automatically find a hundred other examples of a failure they had found.

Creating schemas

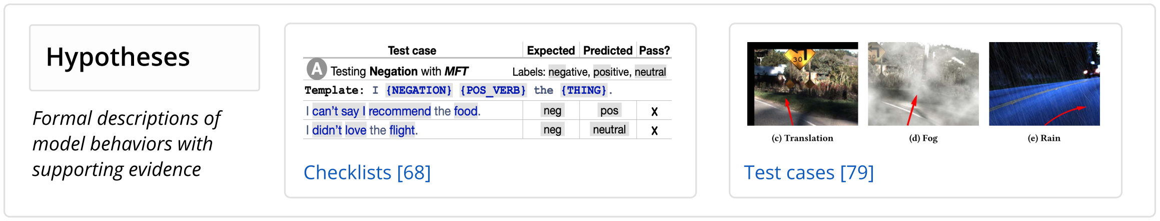
There are myriad tools for creating and visualizing schemas of instances, from faceted layouts [88] to crowd-powered methods for finding areas of high error [83].

Better encodings of classic visualizations such as the confusion matrix can speed up and improve model analysis. For example, unit visualizations showing individual failures allow data scientists to dive deeper into the cause of low performance metrics [45, 87]. Confusion matrices can also be extended beyond binary classification, such as analyzing hierarchical models [101] or comparing multiple models [102].

Novel visualizations can be especially helpful for subgroup analysis. The most direct method is to look at groups of all combinations of features using, for example, data cube analysis [71]. Since this can create a countless number of subgroups, other visual systems allow users to create subgroups from specific features and values [47, 82, 88]. While useful if a data scientist knows what subgroups they want to create, these systems do not lead users towards interesting groupings. Automatic slicing algorithms such as Slice Finder can create a more reasonable number of subgroups with characteristics such as high loss [48]. By slicing data using input features, these visualizations and algorithms create schemas of subgroups highlighting important AI behaviors.

Beyond explicit data slicing, there are also tools for creating schemas of unstructured data. For example, clustering instances can surface semantically similar groups that may have interesting characteristics [84, 103]. Visualizations can also help semantically group data [46]; for example, AnchorVis [89] lets users define "anchors" that spread the data over different semantic dimensions.

Unfortunately, Holstein *et al.* [67] and Wan *et al.* [95] found that knowing *what* groups of instances to create and *how* to group instances are still major challenges for many data scientists. Current schema methods are mostly focused on highlighting known patterns in well-structured domains like tabular data. Additionally, few schema methods help data scientists move on to the hypothesis stage by formally defining hypotheses and gathering diverse supporting evidence. Schema methods that are better informed by hypotheses and can more meaningfully organize large, unstructured datasets could better support data scientists.



Test case	Expected	Predicted	Pass?
A Testing **Negation** with ***MFT*** Labels: negative, positive, neutral			
Template: I {NEGATION} {POS_VERB} the {THING}.			
I can't say I recommend the food.	neg	pos	X
I didn't love the flight.	neg	neutral	X

Figure 4.4: Creating **hypotheses** is the third sensemaking stage. Hypotheses are descriptions of model behavior with supporting evidence. Hypotheses can come from schemas or existing domain knowledge, like checklists and unit tests.

4.3.3 Hypotheses

The third stage of the framework are hypotheses, formal descriptions of model behaviors (Figure 4.4). A hypothesis is a high-level description of a behavior (e.g., *the AI fails in low light*, or *the AI works best for long sentences*) along with supporting evidence. Data scientists test the validity of their hypotheses by gathering enough diverse data to determine how prevalent a behavior is. While hypotheses can come directly from schemas, they can also originate from a data scientist's own domain knowledge or existing behaviors, such as a data scientist experienced with image models checking how a model performs in low-light settings.

Hypotheses in deployed settings are often described as unit or regression tests, well-defined tests of behavior hypotheses [69, 67, 104]. In some cases data scientists even use a test-driven ML approach in which they first define the behaviors that a model should have before training and evaluating the model [100]. For example, participants surveyed by Zhang *et al.* [104] often derive initial behaviors their models should have from specifications of the AI product they are developing. When updating their models, data scientists can check these hypotheses to ensure they are not regressing on important behaviors and monitor any improvements.

Varied external sources can provide hypotheses of model behavior, such as real-world users or customer service personnel. Looking through customer bug reports, customer-facing team members often go through the sensemaking process themselves, finding enough examples of an AI's behavior to describe and report a hypothesis. Hong *et al.* [105] termed the people who find and test these hypotheses "model breakers", roles who interact with customers and may have more direct knowledge of the ways in which a model may behave. From these initial hypotheses, data scientists or testing engineers can go back to the schema and instances stages to collect more evidence and validate the prevalence of reported hypotheses.

Defining hypotheses

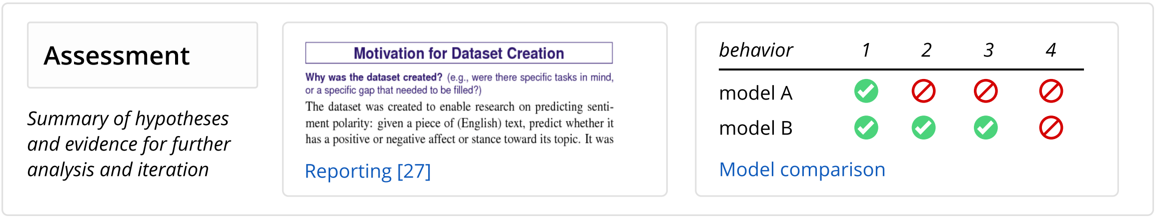
Hypothesis tools help data scientists understand and test model behaviors, especially when tracking multiple hypotheses and assessing supporting evidence.

Visualization systems have shown promise for helping data scientists convert schemas into formal hypotheses. Errudite is a system for NLP models that lets data scientists slice their data into schemas *and* formally define hypotheses of model behavior [46]. Robustness Gym extends this capacity for NLP models by letting data scientists test a variety of hypotheses, from adversarial attacks to data augmentation [94]. There are also systems for statistical hypothesis testing, for example, HypoML is a visual system that lets data scientists statistically test how models perform across specific concepts [90].

Formal testing methods can help scaffold and evaluate hypotheses of model behavior. Even simple checklists of expected behaviors can give data scientists an idea of how well their AI performs in common scenarios [44, 106]. These checklists can be either general descriptions of behaviors or more specific hypotheses with supporting evidence that can validate if an AI shows a behavior. Similar to testing in software engineering, data scientists can also test more low-level behaviors of AI systems [107]. Metamorphic testing, checking if a permutation of an input has an expected impact on the output, can be used to test behaviors such as the impact of weather conditions on a self-driving car [91].

Current tools for creating and testing hypotheses tend to focus on specific, predefined behaviors. They often do not enable data scientists to go back to the schema and instances stages to discover new behaviors and hypotheses. There is also a more limited set of tools for this stage of the process compared to the schema stage. Robust hypothesis creation and evaluation tools could help data scientists more accurately describe and test what real-world behavior their models have.

4.3.4 Assessment


Assessment
Summary of hypotheses and evidence for further analysis and iteration
Motivation for Dataset Creation
Reporting [27]
Model comparison

Why was the dataset created? (e.g., were there specific tasks in mind, or a specific gap that needed to be filled?)
The dataset was created to enable research on predicting sentiment polarity: given a piece of (English) text, predict whether it has a positive or negative affect or stance toward its topic. It was

behavior	*1*	*2*	*3*	*4*
model A	✓	⊘	⊘	⊘
model B	✓	✓	✓	⊘

Figure 4.5: An **assessment** of the model's behavior is the final stage. The assessment provides an actionable summary of a model that can be used for tasks such as improving the model or choosing between different AI services.

Lastly, data scientists combine and organize hypotheses into a cohesive assessment of the behaviors of a model that can be used to make informed decisions (Figure 4.5). For example, when choosing between AI services, a data scientist needs a summary of the models' behaviors to decide which AI provides the best overall performance. Or, in AI development, ML practitioners need to know the most significant failures or areas in which their model can improve the most. Additionally, Yang *et al.* [100] found that ML consultants often report direct data insights and model iterations, assessments, to customers to increase their trust and reliance on a model.

As the most structured stage of the sensemaking process, assessments often act as the starting point for the other AI development processes. For example, a full assessment can be used to decide which AI service is the best for a certain dataset. It can also guide future data collection and model updates to target the areas for which the model performs the worst. Data scientists can then go back to the assessment to see how their updates have changed model behaviors.

AI teams often attempt to track model behaviors to check for serious issues and understand how their AI systems evolve over time. Many data science teams often deal with issues on a case-by-case basis, fixing problems as they are detected in the real world [67]. This introduces its own challenges of ensuring that model updates do not inadvertently regress on certain behaviors while improving others [95]. By having a combined central assessment of model behaviors, data scientists can quickly see their model's overall performance and make informed decisions [44, 94].

Assessment mediums

Recent work has explored how structured reporting about datasets and models can improve future iterations. For example, Datasheets for Datasets [51] tracks the metadata of a dataset, such as provenance and demographic distribution, to inform future model builders, while Model Cards [108] describe AI models to inform their use and potential downsides. Checklists of important steps and processes that data scientists should take can also lead data scientists to more proactively audit the behaviors of their models [109].

Most current assessment tools focus on aggregate metrics and characteristics of a model, whereas AI teams often end up tracking behaviors in an ad-hoc manner. Systems, especially visualizations, that can effectively summarize and track changes in behavior over time could provide a useful and actionable assessment for data scientists. This information can augment documentation methods, for example, with interactive model cards [58], and provide a holistic view of how an AI system is working. While assessment is the final sensemaking stage, it is not the end of the process. Understanding model behavior is an iterative and ongoing process that data scientists continue going through as they update their AI and see new behaviors in the real world.

4.4 AIFinnity System

To assess our framework's *generative* power, we used it to create a system for analyzing image-and-text models called AIFINNITY. AIFINNITY can be used to understand the behavior of a single model using ground-truth labels or compare two models against each other. In the review of existing tools for AI analysis we found that there were a lack of systems that covered the full sensemaking process and helped data scientists move between sensemaking stages. Therefore, our aim was to design a system that met these two goals, using both new and existing AI analysis techniques.

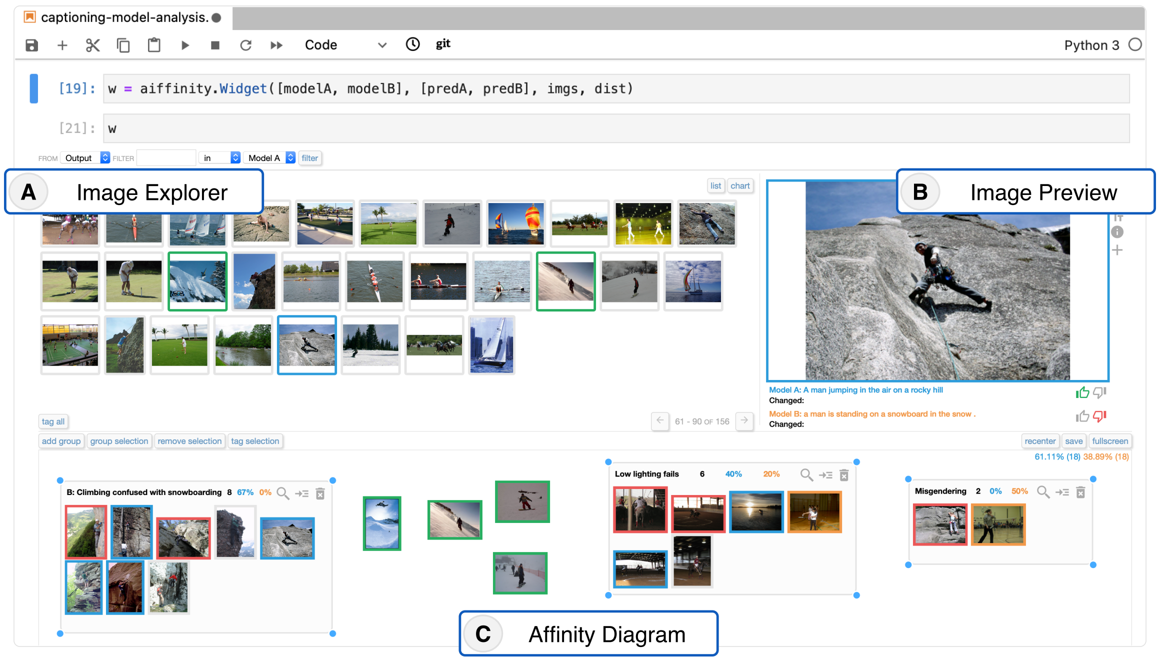
We focused on image-and-text models since they are growing in use for tasks like image captioning, visual question answering, and optical character recognition. Although there are many tools for understanding the behavior of tabular and text models, as described in Section 4.3, there are few tools specifically for image models. Image data is often unstructured, making it difficult to explore instances and create meaningful schemas and hypotheses.

AIFINNITY is a Jupyter widget written in Python and Typescript. Jupyter notebooks are one of the most common data science platforms for data analysis and model training [110]. By making AIFINNITY a widget, we allow data scientists to directly load instances and model outputs from a computational notebook into the tool. AIFINNITY is also model-agnostic, working with common AI platforms such as PyTorch, TensorFlow, and online services.

Running example: optical character recognition

AIFINNITY supports various image-and-text models, but we focus on two primary examples for this work, optical character recognition (OCR) for the system walkthrough and image captioning for the user study in Section 4.5. As a running example of AIFINNITY's workflow, we walk through the example of an AI developer exploring whether their OCR system works for a new dataset of storefront signs [111]. This task is common in real-world scenarios such as Google Maps identifying the names of businesses from streetview data. As we describe AIFINNITY, we use block quotes to describe how a data scientist could use each component in this running example (see Figure 4.7 for an overview):

> Emma is an ML developer at a startup that provides an OCR service. Her company has a new client who wants to use the system to read street signs. Emma is unsure whether their model works for the client's data, so she loads



```
[19]: w = aiffinity.Widget([modelA, modelB], [predA, predB], imgs, dist)

[21]: w
```

Model A: A man jumping in the air on a rocky hill
Changed:
Model B: a man is standing on a snowboard in the snow .
Changed:

Figure 4.6: The AIFINNITY system is a Jupyter Widget that consists of three primary panels, shown here for the image captioning task used in the user study. The **(A) Image Explorer** shows a sample of images, sorted by the images with the most different labels. The **(B) Image Preview** shows the currently selected image. It lets users see the image's extracted metadata, use tools find similar images, and create counterfactuals. Lastly, the **(C) Affinity Diagram** is where users can organize instances into schemas and hypotheses. The colored borders represent the quality judgements from users on whether they believe either or both of the outputs are adequate or not. Data scientists can load AIFINNITY with any AI system and image dataset they are using in a Jupyter Notebook.

AIFINNITY with a sample of the client's storefront images, ground-truth labels, and the AI's outputs. Her goal is to explore how well the AI works for this new dataset to decide whether she needs to collect new data and retrain the model.

4.4.1 Instances, Outputs, and Initial Schemas

AIFINNITY is implemented as a Jupyter widget primarily to enable data scientists to use it with diverse, updating data, directly supporting the **instances and outputs** stage of the sensemaking process. Users pass to the widget a list or two of model outputs and image paths, which can be dynamically updated from the Jupyter notebook. AIFINNITY explicitly supports two outputs for each instance for a couple of reasons. When analyzing a single model, one output can be the output of the AI model, while the other can be ground-truth labels. AIFINNITY can also be used for model comparison, loading the outputs of both models. In both cases, comparing the two outputs provides a useful metadata feature for creating schemas and hypotheses.

The loaded images are displayed in AIFINNITY's image explorer (Figure 4.6A), which shows them in a paginated list. When data scientists hover on a thumbnail or click to select an image, they see the full size version in the image preview (Figure 4.6B) on the right, along with the model output. AIFINNITY initially sorts the instance exploration panel to show instances for which the two outputs are the most different. This creates an initial **schema** or grouping of the data that provides a sensible default for finding interesting hypotheses. When two outputs are significantly different, there is likely some interesting difference between the two. This technique is inspired by common loss functions for NLP models, namely the BLEU score for measuring sentence similarity [112], which we use to calculate how similar two labels are.

As data scientists discover interesting instances, they can drag them to the affinity diagram at the bottom of the interface (Figure 4.6C). Affinity diagrams are a common data analysis tool used in industry and research to organize and track data insights, especially in sensemaking processes [113]. Since images are two-dimensional and humans are especially good at 2D spatial cognition [114, 115], the affinity diagram is a compelling format for spatial organization of images. The affinity diagram serves two primary purposes in the AIFINNITY system, allowing users to create rough **schemas** separate from the image list and to create and track **hypotheses** of behaviors.

> As Emma explores the street sign images in the initial list, she finds that her model does not detect the text in a couple of round signs with text written in a circle. She drags these example images into the same area of the affinity diagram to keep track of them, creating an initial schema.

4.4.2 Schemas With Similar Search and Filtering

Beyond the initial sorted image list, AIFINNITY provides a set of sorting and filtering tools to create new user-defined **schemas**. Since there is no direct technique to explore a dataset of images, unlike queries for tabular data, we provide two complementary features for creating new schemas, similar search and filtering.

AIFINNITY's similar image search enables data scientists to discover instances that may have similar model behaviors. For a selected image in the image preview panel, a data scientist can click on the magnifying glass icon to find the most semantically similar images. Since pixels do not necessarily encode the semantic similarity of two images, AIFINNITY instead uses the outputs of a pre-trained deep learning model to measure similarity. Specifically, AIFINNITY runs each image through the ResNet-18 convolutional neural network (CNN) [116] trained on ImageNet and gets the second-to-last output layer, a 512-dimensional embedding vector representing the semantic content of the image. AIFINNITY then calculates the cosine similarity between the selected image's embedding vector and all other images' vectors in the dataset and sorts the image exploration panel by the most similar images. The data scientist can then drag any interesting images into the affinity diagram. Similar image search acts as a **schema** of instances that are the most semantically similar to a reference image.

> Emma wants to find more examples of round signs with text written in a circle. She selects the first image she found of a round sign and clicks the magnifying glass, which sorts the image explorer to show the most similar images. She finds various images of round signs that her model also fails to detect, so she drags them into the affinity diagram close to the original instance.

While similar image search is a useful heuristic for organizing instances, it is an approximate method that can be biased and miss related instances. AIFINNITY lets data scientists filter images by various semantic features as a more formal way of schematizing the data. When the images are first loaded, AIFINNITY runs two pre-trained deep learning models to extract metadata from the images. First, AIFINNITY gets the ImageNet class of an image using the same pre-trained ResNet-18 model used for the similar image search. AIFINNITY also runs an object detection model (FasterR-CNN ResNet-50 FPN [87]) trained on the MS-COCO dataset to extract common objects from the images. Data scientists can see the extracted metadata for a selected image by hovering over the information button to the right of the image in the image preview. In addition to the extracted metadata, they can also filter images by the labels of either source. For even more control, data scientists can also create custom tags for images that describe any feature of the image.

To filter images by any of these features, data scientists can use the *filter bar* at the top of the interface. Data scientists can use the filter bar to logically combine filters and isolate certain types of instances - for example, a data scientist could filter for images that have a certain object in them but do not have a keyword in the output. As data scientists add filters to the filter bar, the image exploration panel is updated to show only the matching images. Filtering is a **schema** that splits the dataset by explicit semantic features in contrast to similar search's rough grouping.

> Emma hovers over the information button for a round sign with circular text and finds that it is incorrectly classified as an "analog clock". While the class is incorrect, she thinks other round signs may have also been misclassified

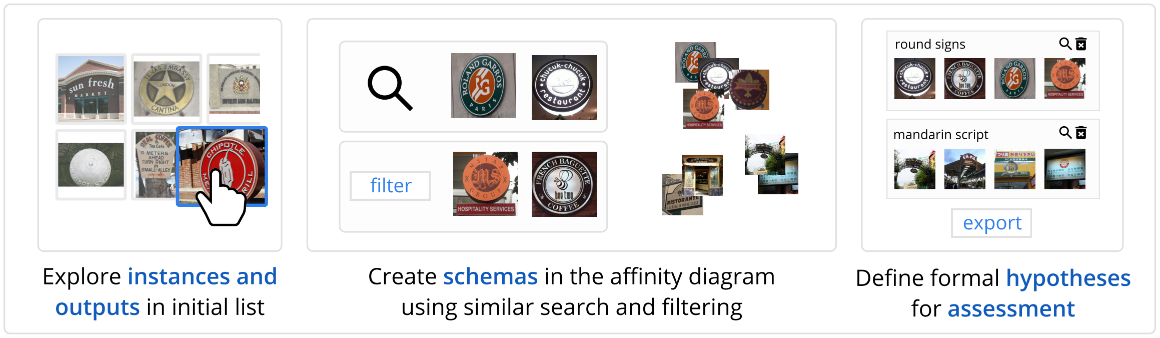

round signs
mandarin script
filter
export
Explore instances and outputs in initial list
Create schemas in the affinity diagram using similar search and filtering
Define formal hypotheses for assessment

Figure 4.7: An overview of the typical sensemaking process used by participants in the user study with AIFINNITY. Participants often started by finding interesting instance in the Image Explorer. They then used the schema tools to find similar behaviors, and dragged them into the Affinity Diagram. Lastly, they created formal hypotheses from the schemas to find more evidence and organized their final assessment. The figure is shown for the optical character recognition task described in Section 4.4

> and decides to filter the images by the class "analog clock." As expected, she finds various other round signs classified the same way, which she drags into the affinity diagram.

These image search and filtering techniques give data scientists multiple ways to schematize and mentally organize their data. From this general organization of images, they can then formulate and validate concrete hypotheses of AI behaviors.

4.4.3 Hypotheses and Assessment

In addition to being a medium for creating schemas of images, the affinity diagram also allows data scientists to create formal **hypotheses** of model behaviors. To create a hypothesis from the schemas, a data scientist can either select multiple images and click the "create hypothesis" button or drag the images into an existing hypothesis. Hypotheses are named rectangles in the affinity diagram data scientists can create for specific behaviors.

The initial evidence used to create a hypothesis is often not sufficient to fully support the prevalence of a behavior. To find more supporting evidence for a hypothesis, AIFINNITY has a modified version of similar image search for hypotheses. When the magnifying glass on a hypothesis is clicked, AIFINNITY calculates the average embedding vector of the images in the hypothesis and sorts the image exploration panel by the most similar images not already in the hypothesis. This allows data scientists to go back to the **schema** stage to find more supporting evidence.

To help data scientists get a more quantitative idea of how prevalent each behavior is, AIFINNITY provides *quality judgements* that can be used to track whether an output is adequate for an instance. For each output on a given instance, a data scientist can indicate whether the output is correct by giving a thumbs up or thumbs down. Each hypothesis then shows the overall percentage of instances for which the data scientist indicated the labels are correct. This gives data scientists a quick quantitative view of how well their AI(s) perform for each hypothesis.

> Emma has dragged various images of round signs into the affinity diagram and decides to create a formal hypothesis. She selects the images, clicks on the *create hypothesis* button, and names the resulting rectangle. To find more evidence, she uses the group similar image search by clicking the magnifying glass on the group. She finds a few more round signs and drags them into the hypothesis. She provides quality judgments for each image in the hypothesis and finds that her AI fails for more than 50% of the signs with circular text.

Since the original dataset may not have enough instances to adequately validate a hypothesis, AIFINNITY also provides a counterfactual feature to allow the creation of more evidence and the refinement of hypotheses. Data scientists can click and drag to draw a black rectangle over an image in the image preview, occluding regions of the image to create a new instance. AIFINNITY then runs the model on the newly modified image and shows the changed text output below the original output. The counterfactual tool allows data scientists to go back to the instances stage and create specific synthetic instances to test their hypotheses.

> Most of the round signs with circular text that Emma found have logos in the center of the circle. Emma is worried that the AI system might actually be failing due to the logo, so she uses the counterfactual tool to create more evidence. She draws a black box in the center of a few of the images to remove the logos and adds the new images as evidence to her hypothesis. She finds that her AI is still not able to detect the text in the new images, further validating her hypothesis.

Lastly, data scientists can organize the affinity diagram with their evidenced conclusions into a final **assessment** of their model behavior, depending on the end goal of the analysis. These insights can then be saved and exported to share with other stakeholders and make actionable decisions.

> Emma organizes the affinity diagram with the main hypotheses she has found, grouping them by the type and prevalence of each behavior. She exports the findings to save the results and uses them to improve her AI's performance for street signs by gathering more data and iterating on the AI's architecture.

4.5 User Study

As a final evaluation of our framework, we conducted an exploratory think-aloud study with 10 professional data scientists tasked with using AIFINNITY to choose between two image captioning models. This study aimed to understand how people use a complete sensemaking system, including how the different stages interact and how data scientists approach the process. We believe that these initial empirical insights can highlight the primary benefits and key features of AI analysis systems grounded in the sensemaking framework.

To recruit participants, we sent an email to 200 data scientists at Microsoft. We continued to invite participants in order of their responses until the qualitative themes in our iterative analysis converged at 10 participants (8 male, 2 female, mean age 32). The participants had an average of 6.8 years of data science experience and worked with various domains and models, including recommendation systems, search, captioning, and cybersecurity. The study lasted between 40 and 60 minutes, for which we compensated the participants with a $25 Amazon gift card.

4.5.1 Study Procedure and Analysis

We started the study with a few background questions about the data scientist's experience with AI and behavioral analysis. The researcher then spent 10 to 20 minutes walking participants through AIFINNITY, specifically for a task comparing two optical character recognition models used to read street signs, the same as the example in Section 4.4. The researcher explained the primary features and components of AIFINNITY, and had the participant create at least one schema and hypothesis. We used a different domain and task for the introduction to not bias the behaviors that the participants looked for in the last part of the study.

In the final and main part of the study, which lasted 30 to 40 minutes, participants were tasked with using AIFINNITY to choose between two image captioning models on a dataset of outdoor activities. This task was motivated by a common use case for image captioning, making photos accessible to people who are visually impaired or blind, for example, on social networks [117]. The task focused on model comparison to give participants a concrete goal, but since comparison requires participants to understand each model's behavior, our discoveries encompass understanding the behavior of one model. The first model, model A, was Microsoft's Cognitive Services image captioning system[1], and the second model,

[1]https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/

model B, was a pre-trained, off-the-shelf captioning model[2]. Participants analyzed the behavior of the models on the UIUC Sports Event dataset [118], a collection of images from various indoor and outdoor sports. We chose this dataset as it has a wide variety of conditions, scenarios, and actions, while being a limited enough domain to explore in 30 to 40 minutes. To not limit or cherry pick the types of behaviors participants searched for, we gave them the general task of understanding the two models well enough to describe to a client, with supporting evidence, which model they should use for the given sports dataset.

As we conducted the studies, we transcribed the recordings and did iterative open coding of the results [119]. We also summarized the schemas and hypotheses of the participants as additional data on how the participants analyzed the two AI systems. With 10 participants, we found that the themes of how data scientists use a complete sensemaking system converged with significantly overlapping interaction patterns and hypotheses. After completing all the interviews, we conducted selective coding of the transcripts focused on the main themes identified in the open coding. We separate the findings into broader insights that are likely to generalize to other sensemaking systems and findings specific to the AIFINNITY system.

4.5.2 Results

Making sense of model behavior

The challenges and goals described by the participants for AI analysis matched those identified in the empirical studies reviewed in Section 4.3. When describing their AI analysis workflow, all 10 participants talked about taking steps to better understand their AI systems beyond aggregate metrics. One participant (P8), a manager of an AI team, actually described their primary role as *"metric development"*: conducting behavioral analyzes on a deployed AI system and converting those insights into metrics to track and improve the system. Another participant (P5) described behavioral analysis as necessary because metrics like *"precision and recall can lie"*, but found that this deeper analysis is *"a very challenging problem."*

Many of the strategies that participants use for AI analysis also reflect those described in the sensemaking framework. Five participants use human judges to label or gather instances, while two participants mostly rely on ad hoc spot checking like dogfooding to check if the AI is behaving as expected. Some data scientists have developed their systems for unit testing and validating model behaviors, with four participants using a form of *"regression sets"* that track specific model behaviors, or hypotheses. They use these

[2] https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning

sets to ensure that updates to their AI do not cause it to regress on important behaviors or subgroups of instances. Even the participants with bespoke tooling found behavioral analysis to be an open challenge, as one participant (P1) stated, *"we don't really have a way of checking for patterns to see if a problem is a one-off or something more systematic."* Like data scientists in the empirical studies, our participants tended to perform behavioral analysis in an ad hoc and post hoc manner, reacting to discovered failures.

Process and strategy

When the participants used the AIFINNITY system, we noticed differences in how participants approached the sensemaking process. The first pattern we found was that participants started the AI analysis process from different stages. Since AIFINNITY does not provide preexisting hypotheses, most of the participants (8) began their analysis by looking at the initial schema of instances with the largest output differences. The other two participants, who train image models in their work, started the analysis with their own preexisting hypotheses. They created these hypotheses from their experience and knowledge of how image models are most likely to fail. For example, a participant (P2) specifically created hypotheses for "high contrast lighting" and "low light" before looking at any of the instances. Despite starting at different stages, all participants eventually took an iterative process, going back to the image explorer to find new instances and using the affinity diagram to create schemas and hypotheses.

Another significant difference in participants' processes was whether they took a breadth-first or depth-first approach. About half of the participants (4) took a breadth-first strategy by exploring multiple instances in the original schema before creating more specific schemas and hypotheses. The other six participants used a depth-first approach, immediately creating schemas and hypotheses for the first interesting instance they found. These different techniques led to a trade-off between the number of hypotheses and the amount of evidence participants found; participants using the breadth-first technique tended to find more hypotheses with less supporting evidence, while depth-first participants found fewer hypotheses with more evidence.

Complementary tools

One of the most salient benefits of having an integrated sensemaking system was the complementarity of tools across stages. As participants progressed through the sensemaking process, they had tools available to help them at each stage. For example, when participants wanted to validate an initial idea of a behavior from a schema, they could create a hypothesis and find evidence using AIFINNITY's similar image search feature. Participants found the progressions between tools and stages to be natural as they created schemas and validated hypotheses.

An unexpected benefit of AIFINNITY was the complementarity of the features *within* each sensemaking stage. This complementarity was most apparent in the schema stage with similar search and filtering tools. The benefit of having both tools was highlighted by one participant (P9), who in validating the hypothesis that models could not describe large groups of people found that *"using the tool together is useful, because otherwise, I was trying to look at [images with] groups of people but [similar image search] didn't give me that, but the object detection model is more specific."* Similar search is a less structured but quicker schema tool, while filtering can create more specific and structured schemas. Participants generally started with the similar search tool to get an initial group of instances for a schema but were concerned about missing evidence with the "black box" search and so moved on to use the filtering approach. Having a quick heuristic tool combined with a more deliberate schema method was an essential feature of AIFINNITY.

Dealing with confirmation bias

Confirmation bias is a significant challenge when creating and validating any hypothesis; How does a data scientist know that they have enough diverse instances to support their hypothesis? We found that having a combined sensemaking system helped data scientists combat confirmation bias. This was especially true when participants went from the hypothesis stage back to the schema stage to find more evidence, as they had various techniques at their disposal to discover or create more evidence. Six of the 10 participants found that at least one of their hypotheses did not hold after finding additional evidence. For example, a participant (P8) thought model A typically confused racquets for video game controllers, but quickly disproved their hypothesis by using the similar image search to find more images of people with racquets that were correctly described. Three participants also actively reflected on their potential confirmation bias and took steps to counteract it by proactively looking for disconfirming evidence.

Actionable, evidenced hypotheses

Overall, the participants found various hypotheses with significant supporting evidence. Participants created 4.1 hypotheses on average, which ranged from specific failures to high-level patterns. The most specific hypotheses included *"model cannot describe images with cliff backgrounds,"* and *"model fails to describe large groups of people on boats."* Some of the most general hypotheses included *"model doesn't describe the central activity," "the model is often too vague,"* and *"bad lighting leads to inaccurate captions."* There was significant overlap in the hypotheses and behaviors the participants discovered despite the wide range of described behaviors. Five of the 10 participants found that Model B confused climbing images with snow, skiing or snowboarding. Four participants found that both models described most of the racquet sports as tennis and did not have *badminton* in their language. Lastly, the most common groupings eight participants created were for a specific activity, for example *climbing*, *boats*, or *tennis*.

At the end of the study, most participants had developed nuanced conclusions about which model they would choose for a given task. The most common conclusion, which seven of the participants came to, was that model A is more conservative, less detailed, but often correct, while model B provides more detailed captions, but is often wrong. Given these findings, they decided to make different recommendations for which model should be used depending on the risk profile and domain of the client.

Beyond describing the differences between the two models, some participants also asked questions about the underlying model and data and came up with potential fixes for the issues they saw. Three participants attributed the patterns they found to biases in the training data or labels. Two of these participants hypothesized that there might be an "alpine" or "snow" bias in the data, causing model B to describe people climbing as snowboarding or skiing, and they wanted to look at the training data to verify their hypotheses. Two other participants hypothesized that the models themselves may be causing the problem by not having certain words in their vocabulary, specifically "badminton" and "croquet", which were often described as "tennis" and "baseball."

Using the AIFINNITY system

We also found insights specific to the AIFINNITY system and analysis of image and text models. Participants generally found the affinity diagram to be intuitive and usable, with five participants specifically stating that it was their favorite part of the interface and one participant (P3) stating that it *"makes total sense, especially for images."* One participant (P8) especially liked the split between the top and bottom areas of AIFINNITY, seeing them as two different representations of the data, or schemas: *"Switching between text and visual representations is very interesting - I can have a hypothesis and go back and forth."* Affinity diagramming is a prolific sensemaking tool in other domains [113], which lends another piece of support to taking a sensemaking lens to AI analysis.

A feature that received mixed feedback in AIFINNITY was the thumbs up or down quality judgement. Two participants (P1, P9) used it as the primary way of tracking which model was performing better, and a third participant (P2) liked that *"having them [images] colored gives you a quantitative feel for how strong your hypothesis is or not."* While more than half of the participants (6) liked to have a quantitative view of their findings, four participants found that the judgement was too coarse to be very useful. Both captions were often wrong, but one was slightly better, or a caption being ‘good’ would depend on the situation. The participants would have liked more detailed descriptions to capture these nuances, such as scale or text descriptions.

Participants thought the counterfactual feature was useful but found that AIFINNITY’s implementation of drawing black boxes was too simple. The participants wanted more image manipulation tools, such as adding new objects or changing image properties. Counterfactuals are a powerful tool for generating more evidence, and participants wanted these improved interactions to test more nuanced and complex behaviors.

4.6 Discussion and Future Work

Through our review of existing studies and tools, the design of AIFINNITY, and the exploratory user study, we found that the sensemaking lens adequately describes how data scientists analyze AI systems. By describing AI analysis using a formal framework, we hope to give researchers and tool creators the language to better understand the context of their systems and studies in data scientists’ overall process. Future work can aim to fill tooling gaps for AI analysis or better understand the challenges and trade-offs in the different sensemaking stages.

4.6.1 Applications and Extensions of AIFINNITY

The think-aloud study focused on one application of AIFINNITY, comparing AI services, but AIFINNITY can be used in various real-world AI analysis scenarios. When working with one model, data scientists can use AIFINNITY to supplement traditional evaluation methods, such as aggregate metrics, by discovering, formalizing, and testing specific model behaviors on a labeled dataset. An example of this process is described in Section 4.4, in which a data scientist tests their model on a new dataset. When using AIFINNITY for model comparison, data scientists can use it on their models, comparing iterations of an AI system using a new architectures or training set.

Participants found the initial set of schemas and hypothesis testing tools to be useful, but additional tools would have to be added to AIFINNITY to make it widely applicable to real-world models. Specifically, there were various behaviors that participants were unable to validate with the current tool set. This was especially the case for issues regarding bias and fairness, which participants wanted to test for but AIFINNITY does not explicitly support. For example, creating schemas across demographic information can help identify potential biases, but the existing metadata did not have those features.

AIFINNITY is primarily an exploratory analysis tool for image models and relatively small datasets that may not generalize to other use cases and domains. The affinity diagramming-based interface can work for other visual data such as videos but may not be adequate for encoding other data types such as text or audio. AIFINNITY also requires users to manually select, explore, and organize instances, which cannot be manually done on a scale of thousands or millions of instances. For formal hypothesis testing on large datasets, especially when comparing models over time, a different system or extensions to AIFINNITY would be required.

4.6.2 Gaps in Current Tooling

In reviewing the current landscape of AI analysis systems, we found a few significant gaps in current tooling. The first limitation is the lack of connection between the "discovery" half (*instances* $\leftrightarrow$ *schemas*) and the "evaluation" half (*hypotheses* $\leftrightarrow$ *assessment*) of the sensemaking process. There are many systems focused on the discovery half of the process that help data scientists slice and explore their data, and many systems for the evaluation half, like checklists and unit tests, that let data scientists validate known behaviors. There are comparatively few tools that let data scientists move between these two processes - turning rough groupings into formal hypotheses or discovering new hypotheses to validate.

There is also a lack of tools for certain types of data domains. Most schema and hypotheses tools are designed for tabular, timeseries, or text data that can be easily sliced and grouped. Unstructured data, such as images and videos, are harder to organize and there exist few usable tools for those domains in most stages of the sensemaking process. With the growing use of image and video recognition in the real world, behavioral analysis tools will be important in detecting and describing their behavior, especially for potential safety or fairness concerns.

4.6.3 Designing and Evaluating Tools With the Framework

The initial patterns found in the user study have some implications for future system design and empirical studies. For example, we found that there is a trade-off between using a breadth vs. depth-first approach when analyzing AI behaviors. A breadth-first approach tends to generate more hypotheses with less evidence, while a depth-first approach leads to fewer hypotheses with more supporting instances. Further experiments or studies could explore whether this leads to disparate insights and whether or not analysis systems should guide data scientists toward balancing these strategies. Other differences in approaches, like starting from certain sensemaking stages, could also be studied to improve data scientist processes.

The process of AI analysis also interacts significantly with other AI tools and processes. For example, explainable AI can be a useful tool at different points in the sensemaking process to both discover and evaluate hypotheses. Model updates and iteration also directly interplay with sensemaking, as people have to make sense of an updated model's new or changed behaviors. These are both complex fields and topics which we did not explore in depth but which interact significantly with understanding AI behavior. Further studies of data scientists and deeper explorations of these interactions could identify areas where tools could bridge or better connect processes, for example, quick feedback loops between model updates and behavioral analysis.

Sensemaking has been applied to domains ranging from organizational psychology to data analysis. Each of these fields has developed unique techniques and tools to improve sensemaking processes that could be used as inspiration for improving AI analysis. For example, there is a growing body of work on *distributed* or *crowd* sensemaking [120, 121], aggregating and reusing schemas and hypotheses from multiple people. Future work could explore how these concepts could be applied to improve AI analysis, for example, reusing common schemas and hypotheses between datasets and models.

4.6.4 Limitations

It is challenging to validate the usefulness of a theoretical framework, and our initial evaluation inherently has some limitations. First, when reviewing existing studies and systems, we likely missed some work that covers stages of our framework or fits the sensemaking process. While we do not claim that we conducted an exhaustive review of the literature, we believe that we covered the major works and subfields of AI analysis rigorously enough to support our framework. Second, to test the *generative* power of our framework, we implemented only one system for the specific domain of image and text models. Although it was not feasible to create multiple sensemaking systems, we believe that the reviewed systems provide a strong foundation for the framework, while the implementation of AIFINNITY serves as a case study of how a complete sensemaking tool can be created. Lastly, our think-aloud study was conducted with participants at one company. While some of their procedures and the insights we derived may have been specific to that company's processes, we chose participants from different teams and suborganizations that act independently in order to increase the generalizability of our results.

4.7 Conclusion

This work introduces a sensemaking framework that describes how practitioners develop mental models of AI behavior. We derived the framework using a sensemaking lens and empirical studies of AI/ML practitioners. We then designed and implemented AIFINNITY, an interactive tool for analyzing image-and-text models, and explored the dynamics of the sensemaking process in an exploratory think-aloud study with 10 professional data scientists. Researchers, designers, and tool creators can use the framework to better understand how people analyze AI systems and develop systems that are grounded in data scientists' analysis process.

CHAPTER 5
IMPROVING HUMAN-AI COLLABORATION WITH BEHAVIOR DESCRIPTIONS

Developers are not the only ones who make sense of AI systems; anyone who interacts with an AI system, such as radiologists who use AI suggestions, develops a mental model of how it behaves. This chapter explores how the resulting information about model behaviors in BDAI can be used to improve human-AI collaboration. We do this by explicitly showing end-users details of model performance on subgroups of data for which the model underperforms. In studies in three domains, we find that these *behavior descriptions* can significantly improve human-AI collaboration. These results emphasize the central importance of model behaviors in understanding and improving deployed AI systems.

This chapter was adapted from my published paper:

> Ángel Alexander Cabrera, Adam Perer, and Jason I. Hong. 2023. "Improving Human-AI Collaboration With Descriptions of AI Behavior". *Proc. ACM Hum.-Comput. Interact. 7, CSCW1, Article 136 (April 2023), 21 pages.*

5.1 Introduction

Human-AI collaboration is finding real-world use in tasks ranging from diagnosing prostate cancer [122] to screening calls to child welfare hotlines [123, 124]. To effectively work with AI aids, people need to know when to either accept or override an AI's output. People decide when to rely on an AI by using their *mental models* [33, 7], or internal representations, of how the AI tends to behave: when it is most accurate, when it is most likely to fail, etc. A detailed and accurate mental model allows a person to effectively complement an AI system by appropriately relying [34] on its output, while an overly simple or wrong mental model can lead to blind spots and systematic failures [7, 125]. At worst, people can perform worse than they would have unassisted, such as clinicians who made more errors than average when shown incorrect AI predictions [126, 127].

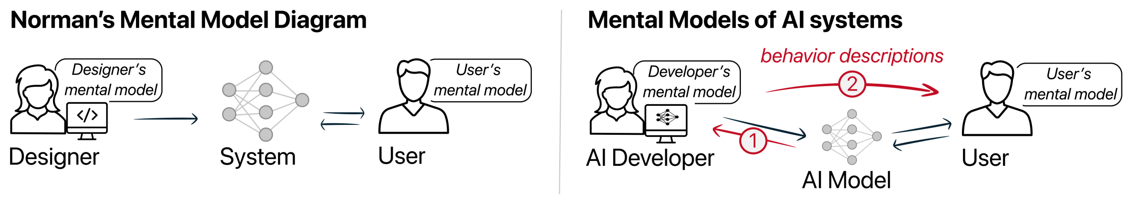


Figure 5.1: Don Norman's mental model framework [8] describes how designers use their mental models to implement systems. End-users then interact with the systems and develop their own mental models of how they believe the systems work. While this process is similar for AI models, a key difference is that an AI is not a direct representation of a developer's intent, but a stochastic model learned from data. This means that **(1)** AI developers *themselves* have to make sense of what an AI system has learned through testing and iteration. Subsequently, they can encode these insights as **(2)** *behavior descriptions*, details of how an AI performs on subgroups of instances, that can be shown to end-users to improve human-AI collaboration.

Mental models are an inherently incomplete representation of any system, but numerous factors make it especially challenging to develop adequate mental models of AI systems. First, modern AI systems are often black-box models for which humans cannot see how or why the model made a prediction [128]. Second, black-box models are also often stochastic and can provide different outputs for slightly different inputs without human-understandable reasons [129]. Lastly, people often expect AI models to behave as humans do, which often does not match their actual behavior [130] and makes people unaware of how an AI may fail [131, 132]. These factors make it challenging for people to develop appropriate mental models of AI systems and effectively rely on them to improve their decision making.

To help people collaborate more effectively with AI systems, we propose directly showing end-users insights of AI behavior (fig. 5.1). We term these insights **behavior descriptions**, details of an AI's performance (metrics, common patterns, potential failures, etc.) on subgroups of instances (subsets of a dataset defined by different features, e.g. "images with low exposure"). Behavior descriptions can take many forms, but should help end-users *appropriately rely* [34] on an AI, using its output when it is most likely correct and overriding it when it is most likely incorrect. The goal of behavior descriptions is similar to that of explainable AI (xAI), but differs in what type of information is provided to end-users. Explainable AI attempts to describe *why* an AI system produced a certain output, while behavior descriptions describe *what* patterns of output a model tends to produce. Thus, xAI and behavior descriptions can be used together to support effective human-AI collaboration.

We hypothesize that behavior descriptions will help people better detect systematic AI failures and improve AI-assisted decision making. Additionally, we hypothesize that people will both trust an AI aid more and find it more helpful when shown behavior descriptions. To test these hypotheses, we conducted human-subject experiments in three distinct domains: fake review detection, satellite image classification, and bird classification. These three domains cover a range of data types, classification tasks, and human and AI accuracies to isolate the effect of behavior descriptions from domain-specific effects.

We found that behavior descriptions can significantly improve the overall accuracy of human-AI teams through two distinct mechanisms. First, for instances with a behavior description, users can directly correct AI failures. Second, people tend to rely more on the AI system for instances without behavior descriptions when behavior descriptions are shown for other underperforming subgroups. We additionally found that showing behavior descriptions had no significant impact on people's qualitative judgements, such as trust and helpfulness, of the AI. Despite the potential benefits of behavior descriptions, their effects are not universal and depend both on how obvious AI failures are to a person and on a person's ability to correct the output once they know the AI is wrong.

In summary, this work introduces **behavior descriptions**, details shown to people working with AI systems of how an AI performs (metrics, common patterns, potential failures, etc.) for specific subgroups of instances. We show how behavior descriptions can improve the performance of human-AI collaboration in three human-subject experiments. These results indicate that knowing the mental models of end-users in human-AI collaboration is essential to understand how a human-AI team will perform and which interventions and decision aids will be most successful.

5.2 Behavior Descriptions

We define **behavior descriptions** as details of how an AI performs (metrics, common patterns, potential failures, etc.) for a subgroup of instances. Behavior descriptions should be semantically meaningful and human-understandable but can vary significantly between datasets and tasks. For domains like binary classification, e.g. spam detection, they can be simple statements of accuracy like *our system incorrectly flags marketing emails as spam 53% of the time.* In more complex domains like image captioning, they can describe specific behaviors like *our system often describes mountain climbing as skiing* or *our system can produce run-on sentences.* The goal of behavior descriptions is to help end-users develop better mental models of an AI and thus *appropriately rely* on the model — using the AI output when it is more accurate and overriding the AI when it is more likely to fail.

In addition to what behavior descriptions contain, *how* they are presented to end-users can vary and impact their effectiveness. For example, *when* behavior descriptions are shown can vary, from only showing them for extreme edge cases to every instance. Similarly, behavior descriptions may only be shown during training or for the first few uses of an AI aid. Due to the broad diversity of potential behavior descriptions, we do not attempt to enumerate all possible forms, and focus instead on testing some core assumptions of their efficacy.

5.2.1 Principles for Effective Behavior Descriptions

To design the behavior descriptions used in this work, we drew from existing studies of human-AI collaboration. From these studies, we derived three properties behavior descriptions should have to maximize their effectiveness. These are not the only principles, but an initial set used to design the behavior descriptions used in this study.

The first property comes directly from the goal of behavior descriptions, to help users appropriately rely on AI output. Therefore, behavior descriptions should provide information that helps end-users decide both *whether* they should rely on an AI and, if the AI is wrong, *how* they should override it. We summarize this principle as **actionable** behavior descriptions that provide end users with concrete information they can act on when using an AI aid.

The second principle comes from studies of AI error boundaries and explainable AI. Bansal *et al.* [7]'s study of people's mental models of AI systems found that models with errors that were *parsimonious* (e.g. simple to define) led to more effective mental models. Separately, Poursabzi-Sangdeh *et al.* [133] found that showing end-users more details about an AI reduced their ability to detect AI errors due to information overload. Thus, behavior descriptions should aim to be **simple**, short, and easy to interpret and remember.

The third and final principle comes from findings on alert fatigue and cognitive load. When alerts or messages are shown continuously, people can suffer from *alert fatigue* and begin to ignore the messages [134]. Additionally, showing people more information increases their cognitive load, which can lead to decreased learning and performance [135, 136]. To avoid these pitfalls, behavior descriptions should be limited and focus on the subgroups of instances with the highest impact. Thus, the third principle is to aim for **significant** behavior descriptions, either common behaviors or those with the most serious consequences.

In summary, the three design principles that we followed to create the behavior descriptions in this work are the following:

1. **Actionable**, suggesting both whether and how a person should override an AI output.
2. **Simple**, aiming to be as parsimonious and easy to remember as possible.

3. **Significant**, limited to behaviors that are common and/or have serious consequences.

5.2.2 Why Not Just Fix AI Failures?

A key question surrounding the utility of behavior descriptions is why developers would not directly fix the systematic model failures or behaviors they discover. Modern AI systems are often stochastic, black-box models like neural networks that cannot be deterministically "fixed" or "updated". ML practitioners interviewed by Holstein *et al.* [67] would often try to fix one problem that would cause the model to start failing in other unrelated ways. In another empirical study, Hopkins and Booth [63] found that practitioners would avoid or limit model updates due to concerns about breaking their models and introducing more failures. Challenges to fixing known model failures are also present in research. This is most apparent with natural language processing models, which are approaching human aptitude in many domains such as question answering and programming [6]. Despite the growing capability of NLP models, many state-of-the-art systems still encode serious biases and harms [137] and fail basic logical tests [44] that developers have been aware of for years, but have not been able to fix.

Fixing model failures can also require significant amounts of new training data, which tends to be expensive and time consuming. Additionally, the specific instances needed to improve certain model failures can be difficult to get, such as globally diverse images or accents [138]. Behavior descriptions can be an important intermediate solution for model issues; end-users can effectively work with an imperfect AI that developers are working to improve and fix.

Lastly, behavior descriptions can be helpful when models are updated. Model updates can be incompatible with end-users' existing mental models and violate their expectations of how an AI behaves, leading to new failures [139]. Updated behavior descriptions can be deployed with a new model to directly update end-user's mental models and avoid decreased performance.

5.3 Experimental Design

To directly test how behavior descriptions impact human-AI collaboration, we conducted a set of human-subject experiments across three different tasks. The tasks range across dimensions such as human accuracy, AI accuracy, and data type to reduce the chance that our results are confounded by domain-specific differences.

5.3.1 Experimental Setup

To test the effect of behavior descriptions, we conducted experiments across three different classification tasks. All three tasks shared the same core setup and only varied in the type of classification task (binary or multiclass) and what participants were asked to label. For each task, we tested three between-subjects conditions to isolate the effect of behavior descriptions:

- **No AI:** Participants were asked to classify instances without any assistance.
- **AI:** Participants were asked to classify instances with the help of an AI they were told was 90% accurate.
- **AI + Behavior Descriptions (BD):** Participants were asked to classify instances with the help of an AI they were told was 90% accurate. Additionally, for instances that were part of a behavior description group (10/30 instances), participants were shown a behavior description stating the AI accuracy for that type of instance (see section 5.3.1 for details about behavior description groups).

Participants in every condition and task were first shown a consent form, introduced to the task, and shown example instances and labels. Those in the condition with the AI were also shown a screen before labeling that introduced the AI and stated its overall accuracy of 90%. The participants were then shown and asked to label 30 instances (see fig. 5.3 for an example UI), 20 instances from the overall dataset, and 5 instances each from two subsets of the data, *behavior description groups*, where the AI performance was significantly worse than for the overall model (see section 5.3.1 for details). After labeling the 30 instances, participants completed a short questionnaire with Likert scale questions, open-ended text responses, and an attention check question.

The experiments were conducted on Amazon Mechanical Turk (AMT) with participants from the United States. We selected participants who had completed more than 1,000 tasks and had an approval rating of more than 98% to ensure high-quality responses. Participants that failed the attention check, a question asking which step they were currently on, were still paid, but were excluded from the results and analysis. Although we considered providing bonuses as an incentive for accurate responses, we found that the incentive to have the task approved was sufficient to get good results without a bonus. We confirmed this by finding similar average accuracy on the control condition for the reviews task, 56%, to that reported by Lai and Tan [140] on the same task using a bonus, 51%.

The study was approved by an Institutional Review Board (IRB) process. We had a total of 225 participants, 25 per task/condition pair. The number of participants per condition

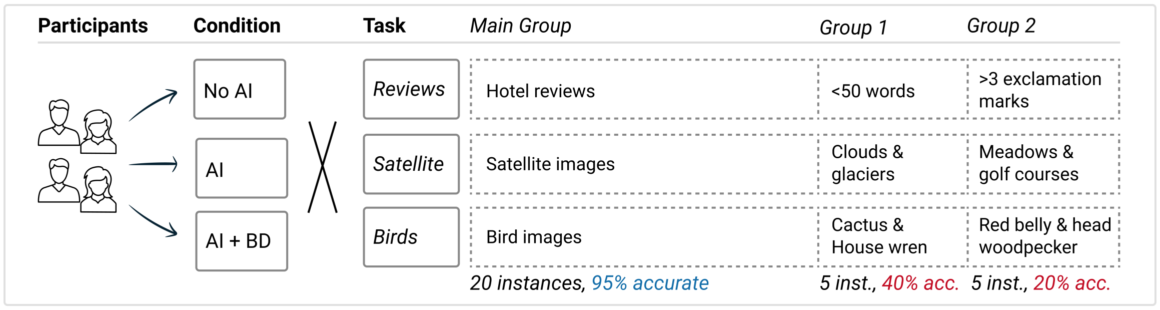


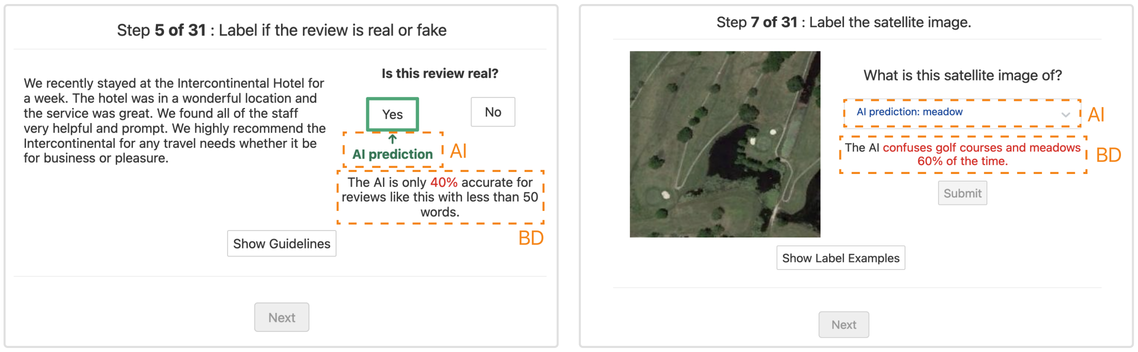
Figure 5.2: Experimental setup. Each participant was randomly assigned to a condition and dataset (3x3 between-subjects study, 25 participants per condition, 225 participants total). For each dataset, participants saw 30 instances, 20 instances from the whole dataset with an AI accuracy of 95%, and 5 instances each from two subsets of the dataset with an AI accuracy of 40% and 20% respectively (simulating subgroups behavior descriptions would be useful for). In the *AI + Behavior Description* condition, participants were shown behavior descriptions for instances in group 1 and 2. While the instances shown to participants were randomly chosen from a larger subset of data, each participant saw the same number of AI errors to ensure they observed the same AI accuracies.

was chosen using a power analysis on the mean and standard error of the accuracy in the initial usability studies for the interface. Of the 225 participants, we removed 13 for failing the attention check. Additionally, we removed three other participants, across two conditions, who had an accuracy of less than 10% (the next highest accuracy being 35%), the same as guessing randomly. We paid participants $2 for the task, which lasted 15 minutes for an hourly compensation of $8 an hour.

Behavior descriptions and wizard-of-oz AI

For this work, we used behavior descriptions stating the model accuracy for subgroups of the data for which the model performs significantly worse than average [47, 83, 48, 44, 141]. Depending on whether the task was binary or multiclass classification, the behavior descriptions resembled text sentences such as ”the model is 20% accurate for this type of instance” or ”the model confuses these two classes 80% of the time”. These types of behavior descriptions are straightforward to create, calculating accuracy on a subset of data, and are actionable for end-users, informing them of how likely it is that they need to override the AI.

In order to control the distribution of instances and AI accuracy each participant saw, we used a wizard-of-oz AI system, mock AI outputs, with a fixed observed accuracy. Of the 30 instances each participant labeled, 20 were randomly chosen from the overall dataset, and the final 10 split into two groups of 5 instances randomly selected from two subsets of the dataset which we term *behavior description groups*. The AI accuracy for instances in the behavior descriptions groups was significantly lower than the overall model to simulate


Step **5 of 31** : Label if the review is real or fake

We recently stayed at the Intercontinental Hotel for a week. The hotel was in a wonderful location and the service was great. We found all of the staff very helpful and prompt. We highly recommend the Intercontinental for any travel needs whether it be for business or pleasure.

Is this review real?

The AI is only 40% accurate for reviews like this with less than 50 words.

Step **7 of 31** : Label the satellite image.

What is this satellite image of?

AI prediction: meadow

The AI confuses golf courses and meadows 60% of the time.

Figure 5.3: UI screenshots for the fake reviews (left) and satellite image classification (right) tasks. Each participant labeled 30 instances, distributed according to the instance groups described in Figure 5.2 and Section 5.3.1. Both screenshots are shown on a labeling step for the *AI + Behavior Description (BD)* condition and on instances that are part of a behavior description group. In the *AI* condition participants are not shown the additional text for instances in a BD group, and in the *No AI* condition participants are not shown the AI output. The bird classification task used the same format as the satellite classification task shown.

the types of subgroups behavior descriptions would be used for. Lastly, while we fixed the distribution of correct/incorrect model outputs per behavior description group, the instances each participant saw were randomized, both sampled from a larger set of images and randomly ordered.

The accuracy breakdown for the subgroups was the following: 95% accuracy (1/20 misclassified) for the *main group* of 20 instances, 40% accuracy (3/5 misclassified) for the first behavior description group, *group 1* and 20% accuracy (4/5 misclassified) for the second group, *group 2*. The behavior description groups for each task are detailed in fig. 5.2 and section 5.3.1. The purpose of the behavior description groups is to have two concrete subsets representing the type of instances for which a behavior description would be used - these are the instances for which we show behavior descriptions in the AI + BD condition.

The accuracy breakdown above gives an actual AI accuracy of 73.33%, not the 90% accuracy stated to the participants, since the task would have been too long to have both an actual 90% overall accuracy and significantly low accuracies for the two behavior description groups. We wanted to simulate a situation where an AI is reasonably accurate, e.g. ¿ 90%, so that a human would want to work with it. Existing work has explored the impact of stated versus observed accuracy and found that there is a small decrease in agreement with the AI the lower the observed accuracy [142]. Since each condition had the same stated vs. observed accuracy discrepancy (90% vs. 73.33% respectively), it should not impact our relative findings on the efficacy of behavior descriptions.

Classification tasks

We chose three distinct tasks for the study to ensure that our findings are not tied to a specific dataset or task. The three tasks vary by data type (text/image), task type (binary/-multiclass classification), and human accuracy (human better/worse than AI). For each task description below, we also detail what types of instances make up *group 1* and *group 2*, the subsets of instances (behavior description groups) for which the AI is less accurate and for which behavior descriptions are shown in the BD condition (see fig. 5.2). The tasks are the following:

Fake Review Detection. The dataset of deceptive reviews from Ott *et al.* [143, 144] has 800 truthful reviews and 800 deceptive reviews for hotels in Chicago. The truthful reviews were collected from online sites such as TripAdvisor, while false reviews were collected from Amazon Mechanical Turk workers. The objective of the task is to determine whether the review is "truthful", written by someone who actually stayed at the hotel, or "deceptive", written by someone who has not. We chose this task since it has been used in previous studies of human-AI collaboration to test the effect of explanations [140] and tutorials [40]. BD groups were chosen from research on common failures in language models:
Group 1 - Reviews with less than 50 words [44].
Group 2 - Reviews with more than 3 exclamation marks [145].

Satellite Image Classification. The satellite images come from the NWPU-RESISC45 dataset [146], which has 31,500 satellite images across 45 classes. The task for the dataset is multiclass classification, labeling each square satellite image with a semantic class. We selected a subset of 10 classes for the task in order to show participants at least a couple instances per class. This task was inspired by real-world human-AI systems for labeling and segmenting satellite images [147]. The BD groups were chosen from areas of high error in the confusion matrices from the original paper [146]:
Group 1 - Cloud and glacier images
Group 2 - Meadow and golf course images

Bird Classification. The bird images came from the Caltech-UCSD Birds 200 dataset [148], made up of 6,033 images of 200 bird species. As in the satellite image task, we chose a subset of 10 classes from the dataset for multiclass classification. The task was inspired by numerous apps and products for classifying bird species [149]. The BD groups were chosen from birds in the same family, classes that are the most similar and difficult to distinguish:
Group 1 - Cactus Wrens and House Wrens

Group 2 - Red Bellied Woodpeckers and Red Headed Woodpeckers

5.3.2 Hypotheses

From the primary goal of behavior descriptions, helping end-users appropriately rely on an AI, we formulated the following hypotheses of how we expect behavior descriptions to affect human-AI collaboration. The hypotheses focus both on quantitative measures of performance and qualitative opinions from participants. Our first hypothesis is that behavior descriptions will improve the overall accuracy of human-AI teams. We hypothesize that behavior descriptions will help end-users more appropriately rely on the AI [34], leading to improved performance.

H1. Showing participants behavior descriptions (BD) results in higher overall accuracy than just showing the AI prediction or no AI.

We hypothesize that this improved performance will primarily come from participants overriding the systematic failures identified by behavior descriptions. By providing actionable descriptions of when the model is most likely to be wrong, for instances in BD groups, we hypothesize that the participants will be able to better identify and correct errors when shown behavior descriptions.

H2. The higher accuracy from showing BDs is due to a higher accuracy on instances that are part of BD groups.

Lastly, we have a set of hypotheses on how we expect people's perception of the AI to change when they are shown behavior descriptions. We hypothesize that participants will find the AI to be *more helpful*, will be *more likely to want to use the AI*, and will *trust the AI more* when they are shown behavior descriptions. These hypotheses come from Yin *et al.* [142]'s study on accuracy and trust in AI systems, which found that observed accuracy significantly affected trust and reliance on AI systems.

H3a. Participants shown BDs trust the AI more than when just shown the AI output.

H3b. Participants shown BDs find the AI more helpful than when just shown the AI output.

H3c. Participants shown BDs are more likely to want to use the AI in the future than when just shown the AI output.

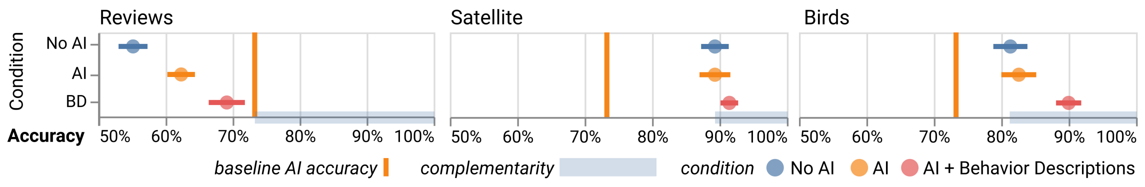


Figure 5.4: Average participant accuracy by task and condition. The vertical orange bar indicates the AI accuracy, what would be the participant's accuracy if they picked the AI response every time. The blue shaded area indicates *complementarity*, the region where the human+AI accuracy is higher than either the human or AI alone. We find that behavior descriptions led to higher accuracy in the reviews and birds tasks, with complementarity in the birds task (red point in rightmost chart). The error bars represent standard error.

5.4 Results

To assess the significance of different conditions on participant accuracy, we used ANOVA tests with Tukey post-hoc tests to correct for multiple comparisons. For the Likert scale questions, we used Mann-Whitney U tests with Bonferroni corrections. Lastly, we used linear models to test for learning effects, also using a Bonferroni correction for multiple comparisons. We used a p value of 0.05 as the cutoff for significance.

5.4.1 Overall Accuracy

To test **H1** we can compare the average human-AI team accuracy for each task across the three conditions. Overall, we found that the *AI* and *AI + behavior descriptions (BD)* interventions have a different effect on team performance in each task (fig. 5.4).

For the reviews task, there was a significant difference in accuracy across the three conditions $(F_{2,64} = 9.18,\ p < 0.001)$. The only significant pairwise difference was between the *No AI* and *AI + BD* conditions $(p < 0.001,\ 95\%\ C.I. = [0.06, 0.22])$. While the AI by itself did not significantly improve the accuracy of the participants, AI supplemented with behavior descriptions led to significantly higher team accuracy, supporting **H1**. Despite the increased performance, there was no human-AI complementarity – higher accuracy than either the human or AI alone – as participant accuracy at every condition was lower than the baseline AI accuracy.

In the satellite classification task there was no significant difference between conditions $(F_{2,67} = 0.63,\ p = 0.534)$. The baseline human accuracy without AI support was the highest across tasks, around 90%, so there was a smaller margin to improve the accuracy of the participants using an AI with a significantly worse accuracy.

Lastly, in the birds classification task, there was a significant difference in participant accuracy $(F_{2,65} = 3.98,\ p = 0.023)$. As in the reviews task, the only pairwise difference

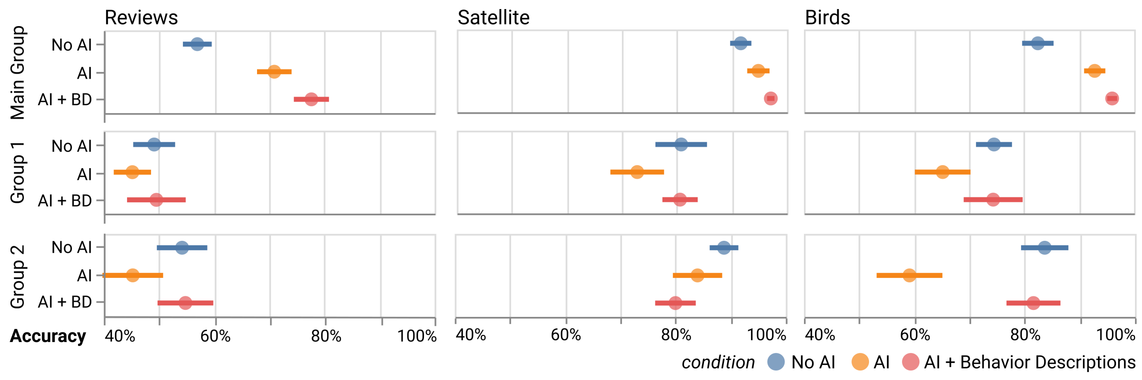


Figure 5.5: Average team accuracy by task, condition and instance group. We further break down accuracy by instance type (see section 5.3.1): the main group (20 instances), and two behavior description groups (5 instances each). The average human-AI accuracy across the three groups of instances gives us an idea of *how* behavior descriptions improve the performance of human-AI teams. We find that participants relied more on the AI when shown BDs in every task. Participant performance on the different behavior description groups was mixed, from no effect to significant improvement in group 2 birds (bottom right). These results highlight the two effects of behavior descriptions, increasing human reliance on a more accurate AI and overriding systematic AI errors. The error bars represent standard error.

was between the *No AI* and *AI + BD* conditions ($p = 0.048,\ 95\%\ C.I. = [0.00, 0.16]$), showing how behavior descriptions can lead to significant increases in participant accuracy using an AI and supporting **H1**. This increased performance also led to complementary human-AI accuracy, higher than that of both the AI or human alone.

In sum, these results **partially support H1**, with behavior descriptions leading to significantly higher accuracy in two of the three tasks. This suggests that while behavior descriptions will not universally improve the accuracy of human-AI teams, they can lead to significant improvements in certain tasks and domains.

5.4.2 Accuracy by Behavior Description Group

To better understand the ways in which behavior descriptions impact performance, we can look at participant accuracy across both conditions and instances in the three different behavior description groups described in section 5.3.1 (fig. 5.5). This allows us to directly test **H2** by seeing if participants in the *AI + BD* condition perform significantly better on the two subsets of instances that have behavior descriptions. The results can be seen in fig. 5.5. We found that this is partially true, as the increased accuracy of the *AI + BD* condition was due both to higher accuracy on the behavior description groups *and* instances in the main group.

In the reviews task, there was only a significant difference between conditions for instances in the normal group ($F_{2,64} = 11.82,\ p < 0.001$), with no differences between conditions for either of the behavior description groups. The *AI* ($p = 0.007,\ 95\%\ C.I. = [0.03, 0.24]$) and *AI + BD* ($p < 0.001,\ 95\%\ C.I. = [0.10, 0.31]$) conditions were significantly more accurate than the *No AI* condition for instances in the main group. These results do not support **H2**, as the higher overall accuracy of the *AI + BD* condition was primarily due to greater reliance on the AI for instances in the main group, not higher accuracy on instances with behavior descriptions.

Despite there being no difference in overall accuracy between conditions for the satellite task, there were differences when looking at the behavior description groups. As with the reviews task, there was a significant difference between conditions for instances in the main group ($F_{2,67} = 5.34,\ p = 0.007$). Participants in both the *AI* ($p = 0.026,\ 95\%\ C.I. = [0.00, 0.09]$ and *AI + BD* ($p = 0.007,\ 95\%\ C.I. = [0.01, 0.10]$ conditions had a higher accuracy than participants in the *No AI* condition for instances in the main group. This is the same result we found in the reviews task, where there were significant accuracy differences for instances in the main group. Despite this difference, the increased accuracy did not translate to a higher overall accuracy for the *AI* and *AI + BD* conditions. Since we did not find any difference between conditions for the two BD groups, these findings do not support **H2**.

Lastly, we found significant differences between conditions for multiple behavior description groups in the bird classification task. As with the other two tasks, there is a significant difference in accuracy between conditions for instances in the main group ($F_{2,65} = 12.22,\ p < 0.001$). Both *AI* and *AI + BD* have a significantly higher accuracy than *No AI*, but there is no significance between *AI* and *AI + BD*. Although there is no significance for instances in group 1, we do see a difference between conditions for instances in group 2 ($F_{2,65} = 6.81,\ p = 0.002$). Both the *No AI* and *AI + BD* conditions both have a higher accuracy than just *AI*. This shows that while participants were able to distinguish between the two types of woodpeckers in group 2 without the AI or when informed about the AI failures, participants trusted the AI and performed significantly worse when just shown the AI prediction. These results support **H2**, as the increased accuracy on group 2 led to higher accuracy in the AI + BD condition.

In summary, these results **partially support H2**. Depending on the task, the higher accuracy in the *AI + BD* condition was due to a higher accuracy on instances that were part of BD groups *and* and a greater reliance on the AI for instances in the main group.

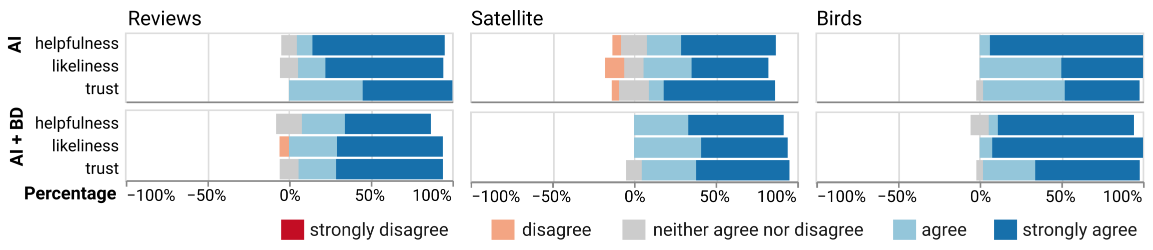


Figure 5.6: Likert-scale responses on perception of AI. The diverging stacked bar chart centered around the neutral response shows that participants across all conditions and subjective measures overwhelmingly viewed the AI favorably. There were no significant differences in user's perception of the AI when they were give behavior descriptions.

5.4.3 Qualitative Results

After labeling the 30 instances, participants were shown a questionnaire page asking about their opinions and feelings of the AI aid using Likert scale questions (fig. 5.6). Since these questions were directly related to the AI output, they were only shown to participants in the *AI* and *AI + BD* conditions. The questions were the following: (1) How **helpful** was the AI for this task? (2) How **likely** are you to use this AI again in a future task? (3) How much do you **trust** the AI? We did not find significant differences between the *AI* and *AI + BD* conditions for any of the Likert scale questions across tasks. These results reject **H3a, H3b, H3c**, and indicate that behavior descriptions do not significantly impact participant perceptions of AI despite differences in how participants use AI predictions.

5.4.4 Additional Findings

In addition to the accuracy metrics and Likert scale responses, we collected and analyzed additional measurements and qualitative responses to further unpack the dynamics of behavior descriptions. Although these are post hoc, exploratory results for which we did not have hypotheses, they can serve as inspiration for further, more formal studies.

One such factor was the time it took participants to label each instance, which can potentially surface interesting insights when compared between conditions and behavior description groups. Unfortunately, the time per instance had high variance and was inconsistent, with numerous outliers. This is likely due to the way AMT workers complete tasks, as they often take breaks or search for new HITs while working on a task [150]. Future studies could incentivize quick responses to gather more accurate time data and measure the speed of participant responses across conditions.

We also tracked in which round each instance was labeled, $(n/30)$, to detect any learning effects (fig. 5.7). To measure learning effects, we fit a linear model of average par-

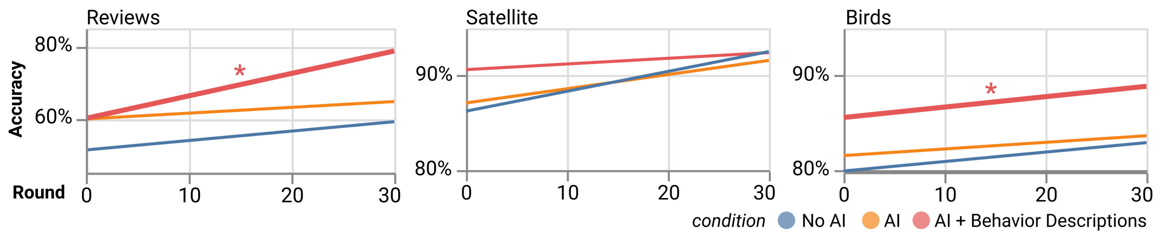


Figure 5.7: Learning curves by task and condition. We fit linear models of accuracy on round number to measure learning effects. We found that the two conditions in which BDs significantly improved performance also had significant learning effects, the *AI + BD* conditions in the reviews and birds tasks (denoted by *).

ticipant accuracy for each condition and each step. We found that for the two domains in which behavior descriptions were effective, reviews and birds, there was also a significant learning effect for the *AI + BD* condition (reviews/AI + BD: $\beta = 0.0062, p = 0.020$; birds/AI + BD: $\beta = 0.0031, p = 0.035$).

The endpoints of the learning curves also show some interesting patterns. For the reviews task, participants in the *AI* and *AI + BD* conditions started at similar accuracies and only over time did participants in the *AI + BD* condition learn to effectively work with the AI and make use of the behavior descriptions. In contrast, in the birds classification task, participants in the *AI + BD* condition were consistently better than the *AI* condition and improved at a similar rate over time. Interestingly, for the satellite domain, the *AI + BD* condition learning curve ends at a point similar to the *No AI* condition but starts significantly higher. This suggests that, while over time participants learned the correct satellite labels or when the AI tended to fail, the behavior descriptions helped bootstrap the learning process.

Lastly, we collected qualitative free response answers from participants about patterns of AI behavior they noticed and general comments they had about the task. As expected, participants in both the *AI* and *AI + BD* noticed that the AI failed in the behavior description groups. While more participants described failures in the *AI + BD* condition than the *AI* condition, the comments were inconsistent and did not show many significant differences between conditions. We found an interesting pattern from comments in the birds task, where participants in the *AI* condition described more general but correct patterns of AI failure. Specifically, a participant found that *"the AI is good at predicting the main class of birds, but might get the sub class incorrect,"* which was the common failure reason between the two BD groups.

5.5 Discussion

These results show that directly informing people about AI behaviors can significantly improve human-AI collaboration. We hope these initial insights spark future work on understanding people's mental models and developing new types of behavior-based decision aids.

5.5.1 Effectiveness of Behavior Descriptions

Overall, the results generally supported our primary hypothesis that behavior descriptions can significantly improve the accuracy of participants working with AI aids. Surprisingly, however, we found that the improved accuracy came from two complementary effects: people overriding systematic AI failures *and* relying more often on the AI for instances without behavior descriptions.

The first effect, helping participants fix systematic AI failures, was the initial goal of behavior descriptions, but we found that it was inconsistent and varied significantly between conditions and behavior description groups. For example, in the reviews task, there was no significant accuracy difference for instances in either behavior description group. This is likely due to the behavior descriptions in the reviews domain not being sufficiently *actionable* and the participants not knowing whether or not to override the AI when they knew it was likely wrong. On the contrary, there *was* a significant difference in accuracy for instances in group 2 of the bird classification task, where participants in the BD condition were able to fix AI failures that most participants with just the AI did not notice. Thus, while we did find some support for this effect, participants fixing AI failures in BD groups was often not the main driver of increased overall accuracy.

In contrast, the more consistent effect was participants in the *AI + BD* condition relying on the AI more often for instances not in the BD groups (main group) compared to the *AI* condition. This increased reliance on the AI when it was more accurate, on instances in the main group, contributed to the higher overall accuracy in the *AI + BD* conditions for the reviews and birds tasks. This was an unexpected effect which was a factor in the higher overall accuracy for participants in the *AI + BD* condition. Although unexpected, the effect is corroborated by studies on trust in AI systems that found that low reliability, AI output that violates people's expectations over time, decreases trust and reliance in an AI [151]. By isolating common AI errors into well-defined, predictable subgroups, the AI appears more reliable to people and can increase their trust and reliance.

Although two of the three domains showed an overall increase in accuracy with behavior descriptions, there was no significant increase in the satellite classification task. We believe that this is due to the high baseline accuracy of humans performing the task, with approximately 90% accuracy, which left little room for improvement with a significantly less accurate AI. The high accuracy of the participants also allowed them to detect and fix the AI errors in the BD groups without any prompting in the *AI* condition. In summary, while behavior descriptions can improve performance, they are not the panacea to human-AI collaboration — AI aids must still provide value and complementarity to human decision makers.

5.5.2 Learning and Behavior Descriptions

We found that participants improved more quickly when using behavior descriptions, learning to effectively complement the AI. The primary pattern we noticed was the significant learning rate in the *AI + BD* condition for the reviews and birds task, as participants quickly learned when they should override the AI. This could be an important property for AI systems that are updated often, as new behavior descriptions could be used to directly update end-users mental models and avoid failures from incompatible updates [139].

A secondary effect we found was a higher initial accuracy in the *AI + BD* condition for the satellite and birds task. While it took time for participants in the *AI* condition to notice how a model tended to fail, the participants with behavior descriptions were aware of the failures right from the start. Even if people's accuracy converges over time, as in the reviews task, behavior descriptions can speed up end-user onboarding and improve early-stage performance.

5.5.3 Authoring Behavior Descriptions

Tools designed specifically for creating BDs could provide important benefits. For example, behavior descriptions do not *have* to be AI failures, but could highlight consistent output patterns or areas where the model performs much better than humans. Bespoke tools could also optimize for creating behavior descriptions with effective properties such as those described in Section 5.2.1. Tools customized to create behavior descriptions could optimize for these different properties.

The types of people who create behavior descriptions can also be much broader than AI/ML developers. With the right tools, stakeholders ranging from quality assurance engineers to domain-specific teams could discover and deploy their own behavior descriptions. In the future, techniques from crowdsourcing could be used to harness end-user's own insights for generating behavior descriptions. This could be, for example, prompting end-users to report consistent failures and patterns that could then be aggregated and voted on [19]. The insights could be processed to automatically generate up-to-date behavior descriptions.

5.5.4 Understanding and Improving Mental Models of AI

These experiments also implicitly tested the more general effect of human mental models on how they collaborate with AIs. We found that when humans have more detailed mental models of how an AI performs, they are more likely to rely on the AI in general. This result is interesting regardless of the use of behavior descriptions, as more experienced end-users will likely develop better mental models of AI aids and gradually change how they rely on the AI.

Developing better techniques for quantifying end-user's mental models of AI systems can help researchers design effective decision aids such as behavior descriptions. This work could take inspiration from studies in HCI and psychology on *cognitive modeling*, mathematical models of human cognition [152]. Cognitive models have been used in education by simulating how people learn math to dynamically teach students [153]. For human-AI collaboration, cognitive models of how people learn AI behavior could inform the design of decision aids such as behavior descriptions.

Our results can also guide the design of machine learning models that can more directly provide behavior descriptions. For example, sparse decision trees could be used to directly generate behavior descriptions to show to end-users. New algorithmic methods that are more amenable to finding clusters of high error could lead to better human-AI collaborative systems.

5.6 Limitations and Future Work

The participants in our study were Amazon Mechanical Turk workers with limited domain expertise in the tasks they completed. Domain experts and professionals, e.g., doctors or lawyers, have deeper expertise in their fields and may develop different mental models of AI systems they work with. For example, they may notice AI failures more often or be less influenced by the information provided by behavior descriptions. Future studies can explore the dynamics of using BDs with domain experts.

Although we selected domains that reflect potential real-world examples of human-AI collaboration, our experiment was a controlled study in a simulated setting. The impact of behavior descriptions may vary when applied to real-world situations with consequential outcomes [38], such as a radiologist classifying tumors. When classification errors have a much higher cost, people may update their mental models differently or act more conservatively when shown BDs.

Our study used one specific type of behavior description, subgroup accuracy. There are countless variations of BDs that can be explored further, such as highlighting subgroups with high accuracy, describing what features of an instance are correlated with failure, or suggesting alternative labels. Future experiments could test these variations to disentangle which features of behavior descriptions are the most effective in improving people's performance.

Lastly, our study focused on the relatively simple domain of classification. Modern AI systems are used in much more complex tasks such as image captioning, human pose estimation, and even image generation. The behavior descriptions for these domains will likely look significantly different from those tested in this work. For example, BDs for a captioning model might focus on grammatical issues and object references rather than statistical metrics. The impact of behavior descriptions will likely vary significantly in these domains, and specific studies could explore both their effect and optimal designs.

5.7 Conclusion

We introduce **behavior descriptions**, details shown to people in human-AI teams of how a model performs for subgroups of instances. In a series of user studies with 225 participants in 3 distinct domains, we find that behavior descriptions can significantly improve human-AI team performance by helping people both correct AI failures and rely on the AI when it is more accurate. These results highlight the importance of people's mental models of AI systems and show that methods directly improving mental models can improve people's performance when using AI aids. This work opens the door to designing behavior-based AI aids and better understanding how humans represent, develop, and update mental models of AI systems.

CHAPTER 6
ZENO: A GENERAL-PURPOSE TOOL FOR AI EVALUATION

To successfully implement BDAI, developers need powerful tools to discover, define, and test behaviors as they iterate on their deployed models. In this section, we introduce Zeno, an interactive tool for discovering and quantifying behaviors for any AI system. Zeno addresses two central sensemaking steps, schemas, and hypotheses, and is an example tool supporting BDAI.

This chapter was adapted from my published paper:

> Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. "Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning." *In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 419, 1–14.*

6.1 Introduction

Machine learning (ML) systems deployed in the real world can encode problems such as societal biases [154] and safety concerns [155]. Practitioners and researchers continue to discover significant limitations and failures in state-of-the-art models, from systematic misclassification of certain medical images [68] to racial biases in pedestrian detection models [156]. In one classic example, Buolamwini and Gebru [13] compared the performance of facial classification models across different demographic groups and found that the models performed significantly worse for darker-skinned women compared to lighter-skinned men.

Discovering and validating model limitations is often termed *behavioral evaluation* or testing [14]. It requires going beyond measuring aggregate metrics, such as accuracy or F1 score, and understanding patterns of model output for subgroups, or slices, of input data. Enumerating what behaviors a model should have or what types of errors it could produce requires collaboration between stakeholders such as ML engineers, designers, and domain experts [53, 18]. Behavioral evaluation is also a continuous, iterative process, as

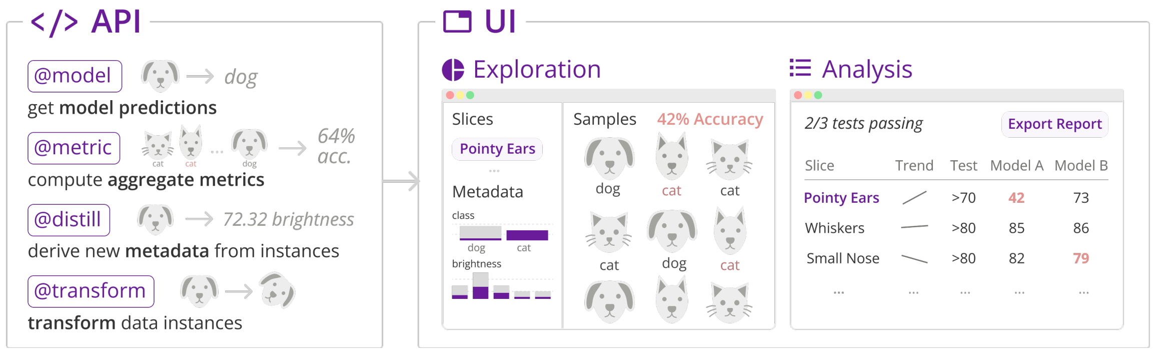


Figure 6.1: ZENO is a framework for behavioral evaluation of machine learning (ML) models. It has two components, a Python API and an interactive UI. The API is used to generate information such as model outputs and metrics. Users then interact with the UI to see metrics, create slices, and write unit tests. In this toy example, a user is evaluating a cat and dog classifier. They see that the model has lower accuracy for dogs with pointy ears, and create a test expecting the slice accuracy to be higher than 70%.

practitioners update their models to fix limitations or add features while ensuring that new failures are not introduced [15].

Despite a growing focus on the importance of behavioral evaluation, it remains a challenging task in practice. Models are often developed without practitioners having clear model requirements or a deep understanding of the products or services in which the model will be deployed [53]. Furthermore, many behavioral evaluation tools, such as fairness toolkits, often do not support the types of models, data, and behaviors that practitioners work with in the real world [157]. Practitioners end up manually testing hand-picked examples from users and stakeholders, making it challenging to effectively compare models and pick the best version to deploy [63].

Given the current state of behavioral evaluation for machine learning, this paper asks two guiding research questions: (1) What are the specific real-world challenges for ML evaluation which are shared across different models, data types, and organizations, and (2) Can an evaluation system addressing these challenges help practitioners discover, evaluate, and track behaviors across diverse ML systems. To this end, we make the following contributions:

- **Formative study on ML evaluation practices**. Through semi-structured interviews with 18 practitioners, we identify common challenges for behavioral evaluation of ML systems and opportunities for future tools.

- **ZENO, a general-purpose framework for behavioral evaluation of ML systems**. We design and implement a framework for evaluating machine learning models across data

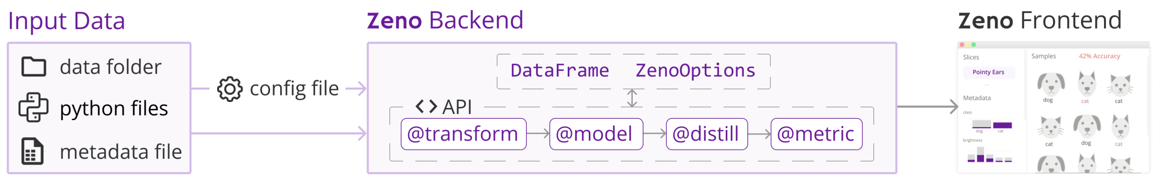


Figure 6.2: ZENO's architecture overview. The ZENO program and inputs (outlined in purple boxes) can either be hosted locally or run on a remote machine. ZENO takes a configuration file with information such as paths to data folders, test files, and metadata and creates a parallelized data processing pipeline to run the decorated Python functions. The resulting UI is available through an endpoint that can be accessed locally or hosted on a server.

types, tasks, and behaviors. ZENO (Figure 6.1) combines a Python API and interactive UI for creating data slices, exportable reports, and test suites.

- **Case studies applying ZENO on diverse models**. We present four case studies of practitioners using ZENO to evaluate their ML systems. Using ZENO, practitioners were able to reproduce existing analyses without code, generate hypotheses of model failures, discover and validate new model behaviors, and come up with actionable next steps for fixing model issues.

6.2 Design Goals

From these interviews and the reviewed studies on ML evaluation, we distilled a set of design goals that a behavioral evaluation system should have. The goals focus on general evaluation challenges identified in the formative study, such as defining behaviors and comparing models. With a system for behavioral evaluation, a user should be able to:

D1. **Evaluate models with different architectures, tasks, and data types.** Machine learning is a broad field with diverse models and tasks ranging from audio transcription to human pose estimation. To reduce the learning curve and encourage the reuse of analyses, users should be able to use one framework to perform behavioral evaluations on most ML tasks.

D2. **Define and measure diverse model behaviors.** Model behaviors are varied and complex, from demographic biases to grammatical failures. Users should be able to encode most of the behaviors across which they wish to evaluate their models.

D3. **Track model performance over time.** Practitioners are continually deploying updated models with new architectures trained on improved data. Users should be able to track performance across models and find potential regressions.

D4. **Evaluate model performance without programming.** Modern machine learning systems are built by large cross-functional teams with nontechnical users. Users should be able to perform behavioral analyses of models without having to write code.

6.3 Zeno: An Interactive Evaluation Framework

We used these goals to design and implement ZENO, a general-purpose framework for evaluating ML systems across diverse behaviors. ZENO is made up of two linked components, a Python API and an interactive user interface (UI). The Python API is used to write functions providing the core building blocks of behavioral evaluation such as model outputs, metrics, metadata, and transformed instances. Outputs from the API are used to scaffold the interactive UI, which is the primary interface for doing behavioral evaluation and testing. The ZENO frontend has two primary views: an *Exploration UI* for discovering and creating slices of data and an *Analysis UI* for writing tests, authoring reports, and tracking performance over time **(D3)**.

Originally, we explored implementing ZENO as either a plugin for computational notebooks or a standalone user interface. We decided on a combined programmatic API and interactive UI as we found it could make ZENO both extensible and accessible. The general Python API allows ZENO to be applied to diverse models, data types, and behaviors **(D1, D2)**, while the interactive UI allows nontechnical users to run evaluation **(D4)**.

ZENO is distributed as a Python program. The Python package includes the compiled frontend which is written in Svelte and uses Vega-Lite [158] for visualizations and Arquero [159] for data manipulation. To run ZENO, users specify settings such as test files, data paths, and column names in a TOML configuration file and launch the processing and UI from the command line (Figure 6.2). Since ZENO hosts the UI as a URL endpoint, it can either be run locally or run remotely on a server with more compute and still be accessed by users on local machines. This architecture can scale to large deployed settings and was tested with datasets with millions of instances (e.g. DiffusionDB [160], 2 million images (Section 6.4.4)).

Running example

To explain ZENO's concepts, we walk through an example use case of a data scientist working at a company deploying a new model. In the following sections, we use block quotes to show how ZENO's features would be used in the example.

> Emma is a data scientist at a startup developing a voice assistant. Her company is using

a simple audio transcription model and she has been tasked with understanding how well the model works for their data and what updates they need to make.

6.3.1 Python API: Extensible Model Analysis

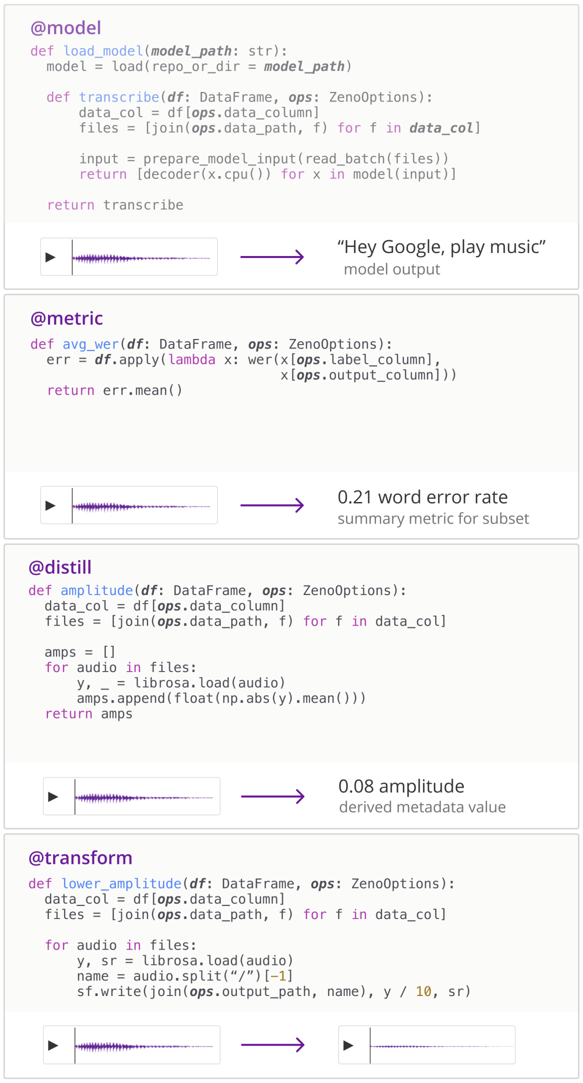
A core component of ZENO is an extensible Python API for running model inference and data processing. The ML landscape is fragmented across many frameworks and libraries, especially for different data and model types. Despite this fragmentation, most ML libraries are based on Python, so we designed the backend API for ZENO as a set of Python decorator functions that can support most current ML models **(D1)**.

The ZENO Python API (Figure 6.3) consists of four decorator functions: `@model`, `@metric`, `@distill`, and `@transform`. We found that these four functions support the building blocks of behavioral evaluation. All four functions take the same input, a Pandas DataFrame [161] with metadata and a `ZenoOptions` object. We chose Pandas as the API for the metadata table due to its popularity, which lowers the learning curve for writing ZENO functions for many data scientists. The `ZenoOptions` object passes relevant information such as column names and static file paths to the decorated API functions. Since ZENO calls API functions dynamically for different models and transformed inputs, `ZenoOptions` is necessary for a function to access the correct columns of the DataFrame.

The two core functions that a user must implement to use ZENO are the `@model` and `@metric` functions. Functions decorated with `@model` return a new function that returns the outputs for a given model. Since this function is model-agnostic, any ML framework or AI service can be evaluated using ZENO **(D1)**. The `@metric` decorated functions return a summary number given a subset of data. `@metric` functions can return classic metrics such as accuracy or F1 score, but can also be used for specific tests such as calculating the percentage of changed outputs after data transformations **(D2)**.

Emma writes a `@model` function which calls her transcription model and returns the transcribed text. She then uses a Python library to implement various `@metric` functions for common transcription metrics such as word error rate (WER).

The two other ZENO decorator functions provide additional functionalities that support behavioral evaluation. Datasets often do not have sufficient metadata for users to create the specific slices across which they wish to evaluate their models. For example, a user may want to create a slice for images with low exposure, but most image datasets do not have the exposure level of an image in the metadata. `@distill` decorated functions return a new DataFrame column for a dataset, extracting additional metadata from instances, and



```
@model
def load_model(model_path: str):
  model = load(repo_or_dir = model_path)

  def transcribe(df: DataFrame, ops: ZenoOptions):
      data_col = df[ops.data_column]
      files = [join(ops.data_path, f) for f in data_col]

      input = prepare_model_input(read_batch(files))
      return [decoder(x.cpu()) for x in model(input)]

  return transcribe
```

```
@metric
def avg_wer(df: DataFrame, ops: ZenoOptions):
  err = df.apply(lambda x: wer(x[ops.label_column],
                              x[ops.output_column]))
  return err.mean()
```

```
@distill
def amplitude(df: DataFrame, ops: ZenoOptions):
  data_col = df[ops.data_column]
  files = [join(ops.data_path, f) for f in data_col]

  amps = []
  for audio in files:
      y, _ = librosa.load(audio)
      amps.append(float(np.abs(y).mean()))
  return amps
```

```
@transform
def lower_amplitude(df: DataFrame, ops: ZenoOptions):
  data_col = df[ops.data_column]
  files = [join(ops.data_path, f) for f in data_col]

  for audio in files:
      y, sr = librosa.load(audio)
      name = audio.split("/")[-1]
      sf.write(join(ops.output_path, name), y / 10, sr)
```

Figure 6.3: The ZENO Python API has four decorator functions: `@model`, `@metric`, `@distill`, and `@transform`. The functions all take the same inputs, a DataFrame and a ZenoOptions object with information such as data paths and column names. `@model` functions return a function for getting running model inference. In the example above, the `@model` function loads a speech-to-text model and returns a function that transcribes audio data. `@metric` functions calculate aggregate metrics on subsets of data. Above, the `@metric` function computes the average word error rate (avg_wer) for transcribed audio. `@distill` functions derive new metadata columns. Above, the `@distill` function calculates the amplitude value from audio. `@transform` functions produce new data inputs. Above, the `@transform` function lowers the amplitude of audio samples.

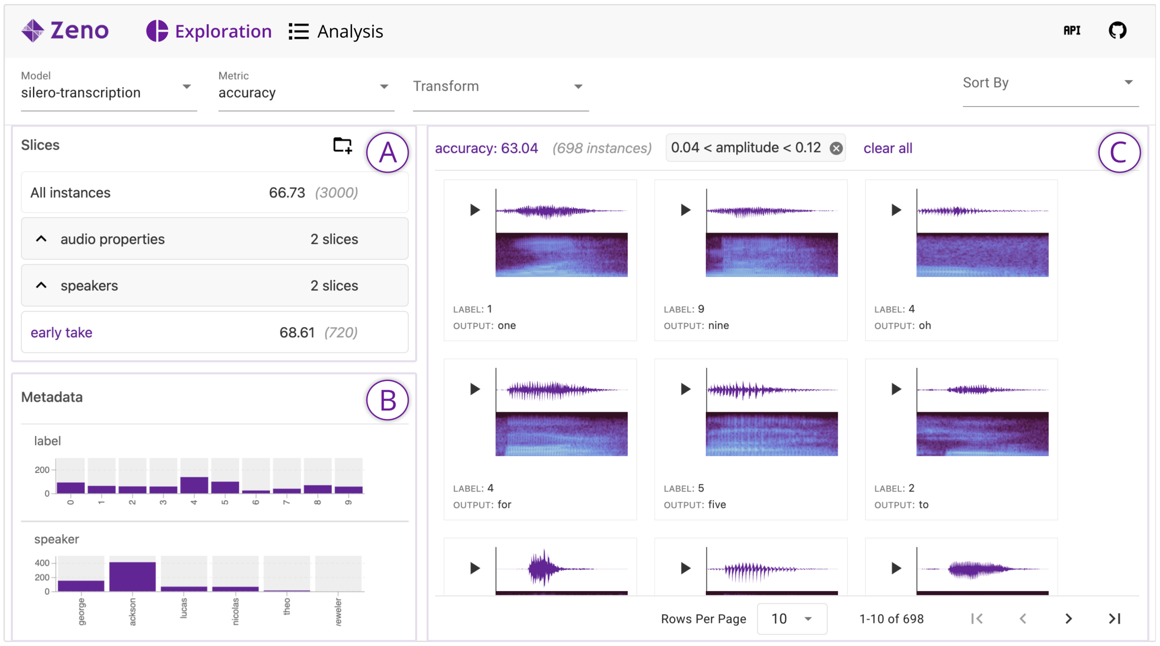
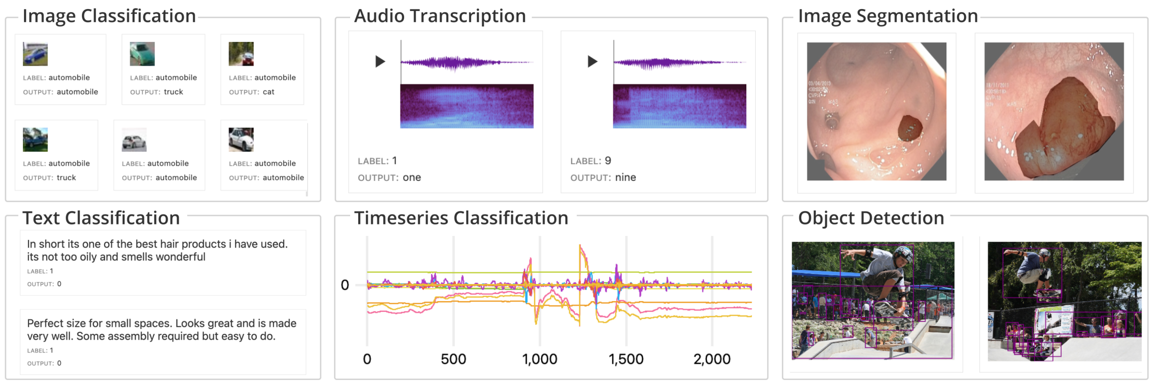


Figure 6.4: The Exploration UI allows users to see data instances and model outputs and investigate model performance. In the figure, ZENO is shown for the audio transcription example described in Section 6.3. The interface has two components, the Metadata Panel (A & B) and the Samples View (C). The Metadata Panel shows the metadata distributions of the dataset (B) and the slices and folders a user has created (A). The metadata widgets are cross-filtered, with the purple bars showing the filtered table distribution. The Samples View (C) shows the filtered data instances and outputs, currently those with *0.04 ¡ amplitude ¡ 0.12*, along with the selected metric, in this case, accuracy.

allowing users to define more specific slices **(D2)**. Users may also want to check the output of their model on modified instances, especially for robustness analyses or metamorphic tests. The `@transform` function returns a new set of modified instances from a subset of instances. For the image exposure example above, a user could write a transformation function that darkens images to check how a model performs for different exposures.

> Emma knows her users have a range of microphones across which she wants her audio transcription model to work well. To test these types of scenarios, she writes a `@distill` function that calculates the amplitude of the sound inputs and a `@transform` function that adds different types of noise.

The ZENO backend builds a data processing pipeline to run the decorated functions and calculate the outputs for the frontend. For example, ZENO parses the code of each `@distill` function to decide whether it depends on model outputs and must be run for each model. Additionally, ZENO runs the processing and inference functions in parallel, which is especially helpful for transform functions, since each `@distill` and `@model`



In short its one of the best hair products i have used. its not too oily and smells wonderful
LABEL: 1
OUTPUT: 0

Perfect size for small spaces. Looks great and is made very well. Some assembly required but easy to do.
LABEL: 1
OUTPUT: 0

Figure 6.5: The instance view of the Exploration UI (Figure 6.4, C) is a modular Python package that can be swapped out for different models and data types. New views can be implemented with a single JavaScript file. ZENO currently has six implemented views, shown here with the following datasets: image classification (CIFAR-10 [162]), audio transcription (Free Spoken Digit Dataset [163]), image segmentation (Kvasir-SEG [164]), text classification (Amazon reviews [165]), timeseries classification (MotionSense [166]), and object detection (MS-COCO [167])

function needs to be run on each transformed instance. Lastly, all ZENO function outputs are cached so any runs after the initial processing are instant.

6.3.2 Exploration UI: Create and Track Slices

To empower nontechnical stakeholders to perform behavioral analyses, the main interface of ZENO is an interactive UI **(D4)**. Although the initial `@model` and `@metric` functions are required to initially set up ZENO, the core behavioral evaluation steps can all be done in the frontend UI by nontechnical users.

The primary tasks in behavioral evaluation are creating subsets of data and calculating relevant metrics. The Exploration page is the initial interface for ZENO and allows users to explore, filter, and create slices of data. It is divided into two sections, the instance view and the metadata panel.

The instance view (Figure 6.4, C) is a grid display of data instances, ground truth labels, and model outputs. Users can select which model output they wish to see, which metric is calculated, and which transformation is applied to the data using the drop-down menus at the top of the UI. A key feature of the instance view is that it is a modular Python package that supports any model and data type **(D1)**. Each view is a separate Python package that implements a JavaScript function to render a subset of data. While views are JavaScript functions, they are packaged as Python libraries so users can install the views they need the same way they install the ZENO package. There are currently 6 views implemented (Figure 6.5), and additional views can be created using a cookiecutter template.

The metadata panel (Figure 6.4, A & B) provides summary visualizations of the metadata columns and previews of user-generated data slices. Each metadata column is shown as a row in the metadata panel, displayed with a different widget depending on what type of metadata it is. ZENO supports 5 main metadata types: continuous, nominal, boolean, datetime, and string. Each metadata widget is interactive and can be filtered to reactively update the instance view and other metadata widgets. When a metadata column is filtered, the filter is shown above the instance view and the selected metric is calculated for the current subset.

When a user finds an interesting or significant subset of data, they can save the current filters as a formal slice. Slices can also be created in the slicing panel, which allows users to visually define and join filter predicates on metadata columns. These slices are displayed at the top of the metadata panel with their size and the selected metric, providing a quick look at the performance for each slice. Users can also create folders to organize their slices.

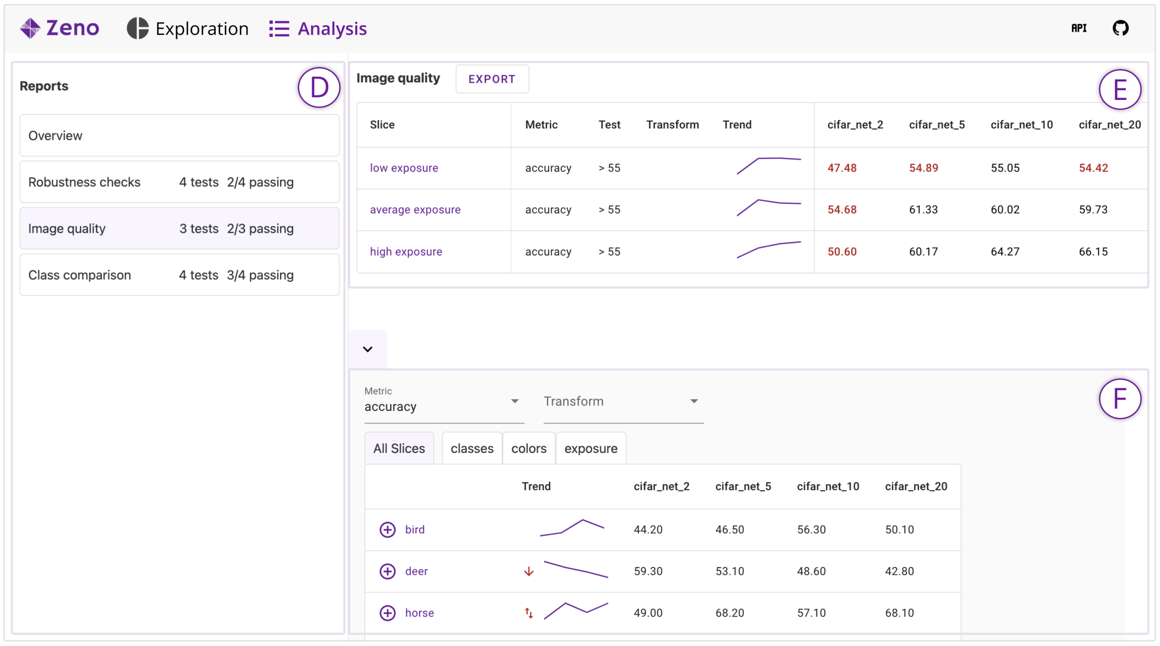
> Emma runs ZENO to analyze her transcription model in the Exploration UI. First, she filters the amplitude metadata widget and finds that the model is significantly worse at transcribing quiet audio. To track this subset, she creates a slice and puts it in the *audio properties* folder (Figure 6.4, A). She then selects the white noise transformation and sees that the error rate increases significantly. She notes that they may want to augment their training data with noisy instances.

6.3.3 Analysis UI: Track and Test Slices Across Models

Once users have created the slices they wish to track using the Exploration UI, they are faced with the challenge of comparing models and slices. The Analysis UI (Figure 6.6) provides visualizations, reporting tools, and testing features to help users better understand and compare the performance of multiple models **(D3)**.

At the bottom of the Analysis page (Figure 6.6, F) is a table showing the slices created in the Exploration page. To help users navigate the slices, folders are shown as tabs above the table and can be used to filter which slices are shown. Users can also select which metric and transform is applied to each slice, and the resulting metric is shown as a column for each model. To make it easier to detect trends in slice performance over time, ZENO shows a sparkline of the selected metric across models for each slice **(D3)**.

A common phenomenon for models deployed in the real world is domain shift, where the real-world data distribution changes over time and model performance degrades [168]. To alert users of potential regressions in model performance, ZENO detects slices with performance that decreases between models. For each slice, ZENO fits a simple linear regression of the selected metric across models, and users are alerted of slices with significant



Reports

Overview		
Robustness checks	4 tests	2/4 passing
Image quality	3 tests	2/3 passing
Class comparison	4 tests	3/4 passing

Image quality EXPORT

Slice	Metric	Test	Transform	Trend	cifar_net_2	cifar_net_5	cifar_net_10	cifar_net_20
low exposure	accuracy	> 55			47.48	54.89	55.05	54.42
average exposure	accuracy	> 55			54.68	61.33	60.02	59.73
high exposure	accuracy	> 55			50.60	60.17	64.27	66.15

Metric: accuracy   Transform

All Slices | classes | colors | exposure

	Trend	cifar_net_2	cifar_net_5	cifar_net_10	cifar_net_20
bird		44.20	46.50	56.30	50.10
deer	↓	59.30	53.10	48.60	42.80
horse	↕	49.00	68.20	57.10	68.10

Figure 6.6: The Analysis UI helps users visualize trends of model performance across slices, and allows them to create *behavioral unit tests* of expected slice metrics. In the figure, ZENO is shown for the CIFAR-10 image classification task comparing models trained for different epochs. The Slice Drawer (F) shows the performance of slices across models, including a sparkline with the metric trend over time. Users can create new reports in the Report Panel (D) and add slices from the Slice Drawer. Lastly, in the Report View (E), users can create *behavioral unit tests* of expected model performance.

negative slope by a downward arrow next to the sparkline **(D3)**. ZENO also highlights slices with high variance, indicating potential unexpected behavior, with a red up-and-down arrow next to the sparkline.

Since domain shift and model updates can lead to unexpected changes in model performance, users may want to set tests for expected slice metrics. We term these *behavioral unit tests*, functions that determine whether a metric for a slice is in an expected range, such as $accuracy > 70\%$. To create tests, users first create a new report (Figure 6.6, D), a collection of slices, and add to it the slices they wish to test. They can then set an expectation for a certain metric on each slice using boolean predicates on the metric value. Models for which the test fails are highlighted in red in the report table, with the overall number of tests that failed for the most recent model shown next to each report in the report panel. Reports can be exported as PDFs to be shared externally from Zeno **(D4)**.

Emma uses the insights from the Exploration UI to train a few new models with new and augmented data. In the Analysis UI she sees that her new models are performing better for noisy input audio, but there is a decreasing trend for instances with lower amplitude.

To ensure that this trend does not continue, she creates a new report and adds slices for different levels of amplitude. She then creates behavioral unit tests expecting each slice to have an accuracy of over 65%.

6.4 Case Studies

We collaborated with four ML practitioners to set up ZENO on models they developed or audited in their work. The goal of these case studies was to answer our second research question, whether ZENO can help practitioners working on diverse ML tasks effectively evaluate their models and discover important behaviors. We chose these case studies as they represented a wide range of tasks (binary classification, multi-class classification, image generation) and data types (text, images, audio), testing how well ZENO generalizes.

Before each study, we met with the case study participant to understand the types of ML systems they use and decide which model(s) they wished to evaluate using ZENO. We then worked with them asynchronously to set up an instance of ZENO, with their model, which they could access on their computer. Finally, we conducted a one-hour study with an interview and think-aloud session (two in-person, two virtual). During the study's first 15-30 minutes, we asked participants about their existing approaches to model evaluation and the challenges they face. For the remainder of the study, participants shared their screen and used ZENO to evaluate the ML model, describing their thought process and findings while mentioning limitations and desired features. Our Institutional Review Board (IRB) approved this as a separate study from the formative interviews. In each of the following sections, we introduce the problem, describe the participant's existing evaluation approach, and detail their findings from using ZENO.

6.4.1 Case 1: UI Classification

For the first case study, we worked with a researcher developing a model to classify smartphone screenshots using a CNN-based deep learning model, which they were evaluating on 10,000 images. The model aims to make UIs more accessible to people with visual impairments by informing them of the type of interface they are looking at. The participant was looking to expand their system to screenshots from other devices, e.g., tablets, and wanted to understand their model's current performance and generalizability. Uniquely for this case study, the participant ran ZENO on a cloud server that hosted their data and models and they accessed the ZENO UI remotely on their laptop.

Existing evaluation approach.

The first participant primarily uses computational notebooks for both *qualitative* and *quantitative* evaluation of their models. For *qualitative* analyses, they select *"some test cases that I hypothesized are hard and easy for the model"*, instances for which they check the model's output to understand how it is behaving. For example, for this model they check a specific screenshot of a login screen with a list structure that they expect the model to misclassify as a list view. For every new domain in which they train a model, the participant spends significant time creating dedicated Python notebooks to display data instances and model outputs for this type of qualitative analysis.

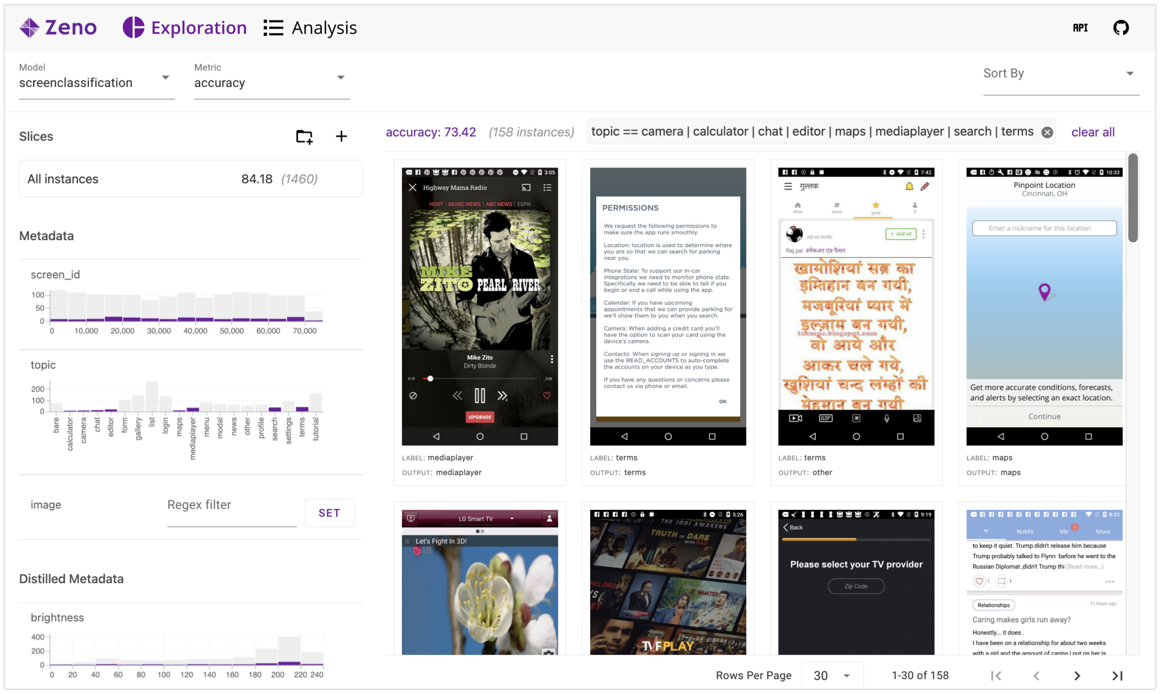
The participant also uses *quantitative* metrics for evaluation, especially for more complex domains such as object detection where they use a combination of metrics such as mean Average Precision (mAP) at different scales. As with the qualitative analyses, the participant authors specific Python notebooks to calculate these metrics. They also make an effort to write evaluation code that is distinct from the training code to ensure that they avoid any bugs such as data leakage in the training process.

Findings with ZENO.

The participant found ZENO's interactive instance view and metadata distributions extremely useful for discovering new failures, systematically validating qualitative analyses, and sharing results with others. Just from the initial Exploration UI, the participant found the ability to quickly browse dozens of instances much more valuable than the static notebooks they used previously. Within a few seconds, they found new model failures they noted to validate later and add as new qualitative test examples. The participant wished to filter the instance view to only see failures or have the system suggest slices to make it easier to quickly find model errors.

With the metadata distributions in the Exploration UI the participant was also able to validate some of their existing qualitative hypotheses more systematically. For example, they confirmed their hypothesis that the model would perform worse for underrepresented classes in the dataset by filtering for the most underrepresented classes using the class histogram (see Figure 6.7). They found the ability to save such slices of data to share with others to be a powerful feature and wished to *"take a very well known dataset such as ImageNet, find slices that are questionable and share them"* to help others test their own model for such issues.

Lastly, the participant found that the code for the ZENO API was similar to what they used in notebooks and that they *"could totally get used to the* ZENO *API"*. While they



Model: screenclassification

Metric: accuracy

accuracy: 73.42 (158 instances) topic == camera | calculator | chat | editor | maps | mediaplayer | search | terms clear all

All instances 84.18 (1460)

screen_id

topic

image Regex filter

brightness

LABEL: mediaplayer OUTPUT: mediaplayer

LABEL: terms OUTPUT: terms

LABEL: terms OUTPUT: other

LABEL: maps OUTPUT: maps

Rows Per Page 30 1-30 of 158

Figure 6.7: A screenshot of the Exploration UI from the UI classification case study (Section 6.4.1). The participant selected underrepresented ground-truth classes and confirmed that the model performance is significantly worse for them.

were able to copy and paste their existing code into ZENO, they wished for a more streamlined setup process, for example, with automatically generated ZENO configuration files for common data types and ML libraries.

6.4.2 Case 2: Breast Cancer Detection

In the second case study, we worked with a researcher who was auditing a breast cancer classification model on a dataset of 6,635 images. The model, also a CNN-based deep learning model, divides mammogram images into small patches and detects whether there is a lesion present in each patch. The model was trained on a dataset provided by a collaboration with clinical researchers at an academic hospital system in the United States. Although the model had a reasonably high accuracy of 80%, the developers had difficulty understanding the failure modes of the model, especially since the dataset was de-identified and had minimal metadata. The participant in our case study wanted to discover meaningful dimensions across which the model failed in order to guide model updates.

Existing evaluation approach

Unlike the first case study participant, the participant in the second study had only used quantitative aggregate metrics when evaluating models. They *"had not used any platform or framework to understand how a model performed on specific features of the metadata"*, and fully relied on aggregate metrics as a measure for model quality. This involved creating Python scripts to load a model and data and calculate metrics such as AUC and F1 score. Attempting to improve the breast cancer classification model led to their first foray into behavioral evaluation.

Findings with ZENO

The participant found that the combination of the extensible `@distill` functions and metadata distributions was essential for finding slices with significant areas of error. Since the participant was not a domain expert, they consulted with medical imaging researchers that recommended a Python library, pyradiomics [169], to extract physiologically relevant characteristics from medical images. The participant implemented dozens of `@distill` functions using pyradiomics functions that encoded important regional information, such as grey-level values, that was not captured by their original features. They also wrote a couple more `@distill` functions to encode the position of each image patch, a hypothesis they had from looking at model failures in the instance view. The participant only had to add a couple of lines of Python to use all of these functions in ZENO.

Since the dataset had minimal existing metadata, interactively filtering the `@distill`ed distributions was the primary way the participant found patterns of failure. By interactively cross-filtering the `@distill`ed metadata histograms, they found that the model performed significantly worse for images with higher tissue density, a phenomenon that also occurs with human radiologists [170]. They also found that the model was trained on many background patches of image that did not include part of the breast, which also impacted the aggregate metrics. The participant noted that they may want to clean the data and upsample instances relevant to the classification task. Due to the quantity and complexity of these analyses, the participant wished for more expressive slice comparisons, such as comparing multiple slices at a time in the Exploration UI. Otherwise, using ZENO the participant found significant failures which they had not been able to find using Python scripts.

6.4.3 Case 3: Voice Commands

The third case study was with a participant who was developing a decision-tree model to detect the direction in which a person is speaking using an array of microphones, which they were evaluating on 11,520 recordings. The goal of the model is to predict to which microphone, often a smart speaker, a person is talking in order to respond from the right speaker. The participant had collected data from diverse setups to understand the performance of their model in the different scenarios.

Existing evaluation approach

Most of the models the participant works on are sensor-based systems highly impacted by the physical nature of the data signals, for example, echoes and noise in sound data. Thus, in addition to calculating classic aggregate metrics, the participant generates and tests inputs with diverse physical properties. For example, in the model described above, the participant collected audio from speakers next to a wall and in the middle of the room to since they thought the rebounding sound from the wall might confuse the model.

To evaluate such scenarios, the participant collects data in dozens of configurations, and so often has extensive metadata for behavioral analysis. Like the other participants, they use computational notebooks to manually split the data across different metadata features and print out multiple metrics. Due to their high quantity of metadata, the participant only looks at simple slices of data, and does not often explore intersectional slices of multiple features.

Findings with ZENO

Using ZENO, the participant was both able to validate all of their hypotheses significantly faster and discovered potential causes for systematic model failures. For example, they confirmed a finding from previous analyses where a *"model worked very well at 1, 2, and 3 meters, but there was a sharp dropoff at 5 meters"* by simply looking at the metadata distributions. They also used the spectrogram visualization of instances in each slice to look for potential reasons for the steep dropoff in performance, for example, signals with lower amplitude. Additionally, they found the cross-filtering between metadata histograms to be useful to find potential interactions between physical features, such as audio both at a distance and a speaker against a wall. Cross-filtering combined with expressive instance visualizations of the audio data was essential for both confirming their hypothesis and ideating potential causes for model failures.

Much of the participant's work is focused on collecting new data, so they suggested data-related improvements for ZENO. Since the participant often tests their model with their own inputs, they wished for a direct way to add new instances to ZENO. They also mentioned having more interactive transformations, for example, having a slider to gradually apply a transformation such as reducing the amplitude of an audio file.

6.4.4 Case 4: Text-to-Image Generation

For our last case study, we worked with a non-technical researcher who explores biases in deployed ML systems, in this case, the text-to-image generation model Stable Diffusion [171]. To audit this model they used ZENO with the DiffusionDB Dataset [160], which consists of 2 million prompt-image pairs generated using the Stable Diffusion model. The participant wanted to explore potential systematic biases in the images generated by Stable Diffusion.

Existing evaluation approach

The participant's work is primarily focused on auditing public-facing algorithmic systems such as search engine results and social media ads. They exclusively conduct manual, ad-hoc audits, testing a range of specific inputs such as search queries and individually checking the model's outputs. The inputs they test are often guided by existing knowledge of model biases, for example, the participant has *"used some lingustic discrimination knowledge [...] such as knowing that certain words tend to be gendered"* to test inputs with likely biased results.

The participant also works with end users of algorithmic systems to understand how they audit models and what biases they are able to find. They found that *"people often found issues in searches that none of the researchers, including me, had even thought of"*. Having diverse users test models is essential for finding issues, and the participant works with end-users to surface new limitations.

Findings with ZENO

When auditing the DiffusionDB dataset with ZENO, the participant took a similar approach to their previous audits but was able to come up with more systematic and validated conclusions of model biases. Their primary interaction with ZENO was using the string search metadata cell to look for certain prompt inputs. Similar to how they approached debugging search engines, they used prior knowledge of likely biased prompts but were able to see dozens of examples instead of one prompt at a time. For example, when searching for prompts with the "scientist" in them, every generated image was male, encoding a typical gender bias. By seeing dozens of prompts the participant was able to gather more evidence that the model produced this pattern systematically and was not due to a one-off prompt.

The DiffusionDB dataset also includes a measure of toxicity, or "NSFW" level, for both the input prompts and generated images. These numbers were represented as histogram distributions in ZENO, and the participant found it invaluable to filter by and find potential biases. One interesting experiment the participant tried was to see if the average distribution of the NSFW tag would go up for certain terms. For example, they saw small increases in the distribution when searching for certain gendered terms, including the word "girl", which reflected that the images generated of women were more sexualized than those of men. They could only see this dataset-level pattern using the combination of ZENO's metadata distribution and instance view.

Lastly, the participant reflected on how usable ZENO would be for everyday users of algorithmic systems. They mentioned that technical terms such as "metadata" may be too niche for everyday users and could be renamed. Otherwise, they found the system intuitive and usable if set up for use by diverse end users.

6.5 Discussion

Our case studies showed that ZENO's complementary API and UI empowered practitioners to find significant model issues across datasets and tasks. More generally, we found that a framework for behavioral evaluation can be effective across diverse data and model types **(D1)**. This generalizability can be seen by comparing two of the case studies, the malignant tumor detection (Section 6.4.2) and audio classification (Section 6.4.3) cases. The two cases differed significantly in their data type (image vs. audio), task (binary vs. multi-class classification), model (CNN vs. decision tree), and end goal (model development vs. auditing). Despite these differences, both participants could effectively discover and encode model behaviors they wished to test and found limitations ranging from robustness to domain shift **(D2)**.

ZENO's different affordances made the behavioral evaluation process easier, quicker, and more effective, depending on the user's goals and the challenges of each particular task. For example, in Case 2, the participant found the extensible API essential for creating metadata to analyze their model across **(D2)**, while in case 3, the participant found the interactive visualizations more useful given the extensive metadata already present in their dataset. ZENO also supports users' particular strengths and skillsets - without using the API, our non-technical case study participant (Case 4) was still able to find significant model biases by using their domain knowledge to interact with the UI **(D4)**.

Participants in the case studies found that ZENO was easily integrated into their workflows, requiring minimal effort to adapt their code to work with the ZENO API **(D1)**. For example, the participant in case study 1 only modified a few lines of their inference code to work with ZENO, and the participant in the second case study was able to use a radiomics library in ZENO with minimal setup. The participants also suggested ways in which ZENO could be made even easier to use, such as automatically generating ZENO API functions and configuration files for common ML libraries.

While we validated that most of the design goals were met by ZENO, our case studies did not thoroughly explore how ZENO could be used over longer periods **(D3)**. All four participants worked with early-stage models and only used ZENO for a limited time. Longer-term, in-situ studies would provide more nuanced feedback for the utility of ZENO's model comparison features. A benefit of ZENO's ease of use, both with the API and UI, is that users can immediately start using ZENO's model tracking and comparison features as models move from research to deployment.

6.6 Limitations and Future Work

ZENO provides a general and extensible framework for the behavioral evaluation of ML, but leaves significant room to better address the challenges in the evaluation process.

Slice discovery. A central challenge for behavioral evaluation is knowing *which* behaviors are important to end users and encoded by a model. To directly encourage the reuse of model functions to scaffold discovery, we are currently designing *ZenoHub*, a collaborative repository where people can share their ZENO functions and find relevant analysis components more easily. Including slice discovery methods directly in ZENO could also help users find important behaviors. ZENO provides the common medium of representing metadata and slices that practitioners can use to interact with and use the results of these discovery methods.

Improved visualizations. Defining and testing metrics on data slices is the core of ZENO, but it only provides a few simple visualizations of data and slices in a grid and table view. There are many more powerful visualization types that could improve the usability of ZENO. Instance views that encode semantic similarity, such as DendroMap [172], Facets [88], or AnchorViz [89], could improve users' ability to find patterns and new behaviors in their data. ZENO can also adapt existing visualizations of ML performance, such as ML Cube [71], Neo [101], or ConfusionFlow [102], to better visualize model behaviors. For example, grid views showing the intersections of slices could highlight important subsets of data.

Scaling. ZENO has a few optimizations for scaling to large datasets, including parallel computation and caching, but machine learning datasets are continuously growing and additional optimizations could speed up processing considerably. A potential update would be to support processing in distributed computing clusters using a library such as Ray [173]. Another bottleneck is the cross-filtering of dozens of histograms on tables with millions of rows. ZENO could implement an optimization strategy like Falcon [174] to support live cross-filtering on large datasets.

Model improvement. ZENO is focused exclusively on *evaluation* and does not include methods to update models and fix discovered failures. Future work can explore how to directly use the insights from ZENO to improve model performance. For example, there are promising results in using data slices to improve model performance, such as slice-based learning [175] and group distributionally robust optimization (GDRO) [176, 177].

Further evaluation. The case studies evaluated ZENO on real-world ML systems, but further evaluations could better elucidate the affordances and limitations of ZENO. Future evaluations could explore how usable ZENO is for additional non-technical users and how well it works for continually updated deployed systems.

6.7 Conclusion

Behavioral evaluation of machine learning is essential to detect and fix model behaviors such as biases and safety issues. In this work, we explored the challenges of ML evaluation and designed a general-purpose tool for evaluating models across behaviors.

To identify specific challenges for ML evaluation, we conducted formative interviews with 18 ML practitioners. From the interview results we derived four main design goals for an evaluation system, including supporting comparison over time and no-code analysis. We used these goals to design and implement ZENO, a general-purpose framework for defining and tracking diverse model behaviors across different ML tasks, models, and data types. ZENO combines a Python decorator API for defining core building blocks with an interactive UI for creating slices and reports.

We showed how ZENO can be applied to diverse domains through four case studies with practitioners evaluating real-world models. Participants in the case studies confirmed existing findings, hypothesized new failures, and validated and discovered behaviors using ZENO. As a general framework for behavioral evaluation, ZENO can incorporate future features, such as error discovery methods and visualizations, to support the growing complexity of models and encourage the deployment of responsible ML systems.

CHAPTER 7
ZENO REPORTS: AUTHORING INTERACTIVE AND REPRODUCIBLE AI EVALUATIONS

The Zeno platform is a powerful tool for discovering and quantifying the behaviors of AI systems. Despite this, Zeno does not cover every step of the sensemaking process - for the last step of *assessment*, users have to leave Zeno and create summary reports manually. In the last chapter of the thesis, I introduce an authoring tool for *Zeno Reports*, which allows practitioners to create interactive, shareable, and reproducible evaluation reports directly from their Zeno analyses.

7.1 Introduction

General-purpose AI models like GPT-4 [178] are making it easier for anyone to create complex AI applications such as programming assistants [2] and augmented writing platforms [179]. Understanding the capabilities and limitations of a model is essential to iteratively create, improve, and debug these systems. This process is especially important for societally impactful behaviors such as biases and safety concerns. Numerous techniques and tools have been proposed for this type of AI analysis, ranging from algorithmic error discovery methods [49, 48, 84, 85] to visual analytics tools [47, 46, 54]. Zeno, described in the previous chapter, is a tool that generalizes this AI analysis process into one platform that works across data and model types.

Despite this progress, there remains a significant gap between the analysis tools and the tools to *communicate* their results [180, 181]. Engineers building models, auditors conducting due diligence, and other stakeholders involved in model development must communicate their results to other stakeholders and downstream users to facilitate a model's development and appropriate use. The results tend to be summarized and shared through media that is disjoint from the tools used to perform the analysis [51]. Transferring the results into common mediums like PDFs requires re-doing a significant amount of work, and the results quickly become outdated when the model is updated. Practitioners interviewed by Deng *et al.* [182] found that creating accessible analyses, visualizations, and documentation was *"extremely time consuming"* and often not recognized by colleagues. Additionally, these static reports are often hard to reproduce in new models and lack important communication features such as interactive visualizations.

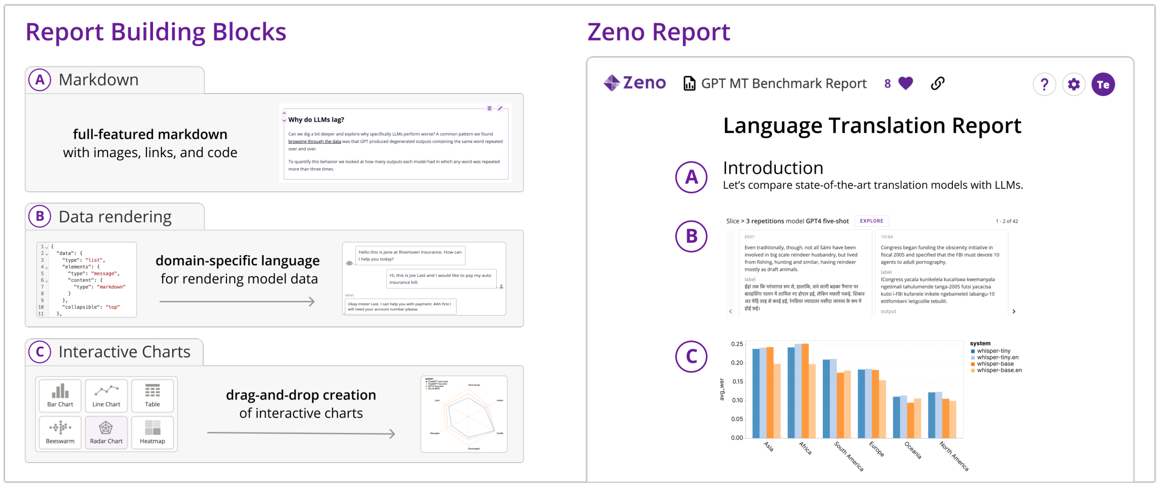

Report Building Blocks
A Markdown
full-featured markdown with images, links, and code
B Data rendering
domain-specific language for rendering model data
C Interactive Charts
drag-and-drop creation of interactive charts
Bar Chart
Line Chart
Table
Beeswarm
Radar Chart
Heatmap
Zeno Report
Zeno
GPT MT Benchmark Report
Language Translation Report
A Introduction
Let's compare state-of-the-art translation models with LLMs.
B
C

Figure 7.1: We introduce an authoring tool for *Zeno Reports*, visualization-driven analyses of AI system behavior. *Zeno Reports* are notebook-style reports composed of three core building blocks: *Data*, *charts*, and *markdown*. *Data* cells are specified using a domain-specific language to render any input data and model output, from audio to images and text. *Chart* cells are interactive visualizations created using a visual editor. *Markdown* cells contain text for structuring and adding narrative to reports. *Zeno Reports* can be authored collaboratively and shared directly through the platform. The reports can also be automatically updated with results for new models, improving the reproducibility of analyses.

To improve the authoring experience and delivery of AI analysis reports, we take inspiration from the area of *narrative visualization*. *Narrative visualizations* are textual stories supplemented by interactive visualization to tell data-driven narratives. There is a rich literature on authoring tools for creating these visualizations, but they generally require significant programming skills, visualization design knowledge, or learning a domain-specific language. These requirements are a barrier to entry for many potential authors of AI analysis reports, as they are often created collaboratively with authors with varying programming and visualization proficiency. Furthermore, these authors typically cannot invest the time required to learn a new language for report authoring as this is typically only a part of their day-to-day responsibilities.

In contrast to the full design space of any data visualization, AI evaluations can almost always be distilled into the narrower design space of metrics on specific data slices, which is the insight we used to build Zeno. A *slice* as a subset of a dataset defined by some metadata (e.g. $age > 10$ & $birthplace == spain$) and a *metric* as a number calculated on an instance and averaged over a slice (e.g., accuracy). We use this insight to design and implement a visual authoring tool for creating *Zeno Reports*, interactive, reproducible evaluations of AI models. *Zeno Reports* can be authored in a fully visual GUI and are collaborative, interactive, and shareable. They are made up of three core components.

Data components can render any input and output data type, from audio to images. *Chart* components are created in a full visual editor and support six common visualizations for AI evaluation. Lastly, *markdown* cells provide commentary around the data and charts, supporting the full set of markdown features, such as images, videos, and code.

To evaluate the effectiveness and utility of *Zeno Reports* for various users, we explore its efficacy in two domains. First, we deployed *Zeno Reports* to the general public and had over 500 users create over 200 reports for diverse tasks and data types. We also worked with a group of AI experts to conduct an in-depth evaluation and comparison of state-of-the-art language models [183]. We found that novices and experts could craft compelling and insightful narratives of AI behavior in various AI tasks using *Zeno Reports*. While we found some interesting differences in how the two groups authored reports, such as experts who rely on charts more than text to show their findings, all users primarily relied on simple bar and line charts supported by text to communicate their findings. These insights can guide the development of future AI reporting tools, which would benefit from focusing on discovering high-impact AI behaviors rather than the visual encoding of the insights.

In summary, our technical contributions are as follows.

1. A **no-code platform** for authoring *Zeno Reports*, interactive AI evaluations. Users can combine markdown, data, and chart building blocks to create interactive analyses describing the nuanced behaviors of AI systems.

2. A **domain-specific language** (DSL) to define how AI inputs and outputs are rendered. The DSL defines data rendering elements (e.g., images, audio, 3D, etc.) and display types (list, map, etc.) to support displaying complex and/or multimodal data.

3. A **visual chart editor** for creating visualizations specific to AI evaluation and analysis. The chart authoring experience is tailored to the core analysis dimensions of the slices, metrics, and models.

7.2 Design Goals

We used insights from three diverse areas to establish the design goals for an authoring tool for AI reports - both for the content reports should support and the authoring process. First, we examined the specifications of existing AI reports to identify the essential information that reports should support. Second, we used the design guidelines of a study on reporting needs in business intelligence applications [184] to understand what features a report authoring tool should have. Third, we explored how narrative visualization techniques can enhance reader comprehension.

We define an “AI report” as an artifact detailing any aspects of an AI system, such as its technical implementation, training process, evaluation, behaviors, etc. Myriad specifications have been proposed for specific types of reports. For example, Model Cards [51] is a widely adopted specification defining common information that should be reported when a model is released. FactSheets is a similar specification inspired by *supplier’s declarations of conformity (SDoCs)* in other industries. Although these specifications are for formal reports created by developers for external publication, there are other forms of reports for internal development, auditing, and due diligence. Checklist by Ribeiro *et al.* [44] runs logical tests on language models and produces a capabilities report of behaviors used by developers to fix model errors. Developers also create one-off internal analyses to share with other internal teams with insights such as potential biases [182]. Despite their varied audiences and purposes, all reports share common text, charts, and instance building blocks.

Empirical studies of how AI teams collaborate have discovered common limitations in how reports are authored and shared. Importantly, most of the reporting literature describes *specifications* of what reports should have, but does not help users implement reports. Crisan *et al.* [58] and Deng *et al.* [182] found that the creation of reports is laborious and that the resulting reports often missed important details and information such as “unintended uses or disaggregated model performance” [58]. Additionally, the resulting static report cannot be used for further analysis, so those who want to reproduce or extend the results must recreate their analyses. Lastly, reports are disjoint from the underlying analysis code and have to be manually updated or augmented with new results.

Existing specifications tell us *what* information a report should have but don’t tell us what the authoring experience should be like. To define the features that an authoring platform should have, we used insights from a study by Zhang *et al.* [184], which interviewed 15 business analysts about their report authoring needs. The authors summarized their findings in a set of design guidelines that we integrate into our design goals: GUI-based authoring, interactivity in charts and visualizations, and the inclusion of supporting text narratives. Although business intelligence is a different task than AI analysis, we found when developing the sensemaking framework in section 4.3 that there are significant parallels between the two processes, supporting the applicability of these guidelines for model report authoring.

Lastly, we use design concepts from the field of narrative visualization to determine the output design of a reporting platform [59]. Reports should primarily be author-driven to cover the core model details and behaviors described by specifications such as model cards. They should still be interactive and support a martini glass narrative structure [59] that begins with a primarily author-driven narrative and then opens up to user interaction. This ensures that core insights delineated by frameworks like Model Cards can be incorporated while allowing readers to explore the data and conduct their analyses.

Using these insights from the existing literature, we derived the following design guidelines for an AI report authoring tool.

D1. **Report building blocks supporting core analyses.** Model analyses require text describing important details, charts quantifying performance, and visualizations of input and output data. A report-authoring platform should support both creating and rendering these elements.

D2. **Live results from underlying data.** Report authors should be able to convert their model analyses directly into final reports. Direct linking analysis with reporting significantly reduces the effort required to write reports while ensuring that details are not missing or misreported. Reports should support a "static" mode showing a specific snapshot of data and a "live" mode that is updated with the underlying data.

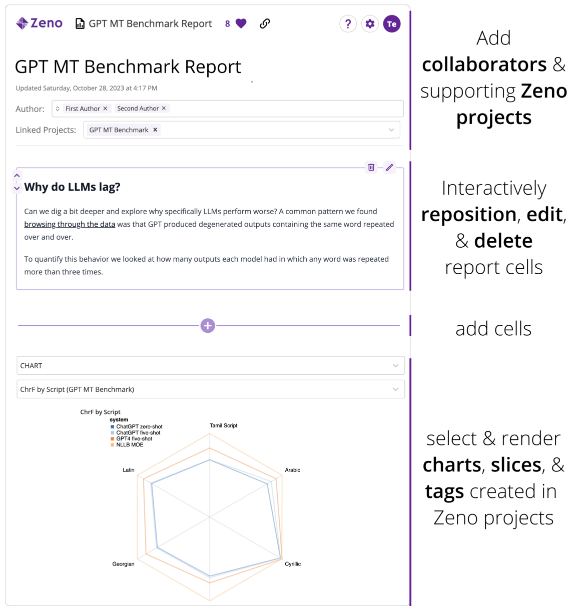
D3. **Interactive elements to support extending and reproducing analyses.** Report elements should be interactive and explorable. Readers should be able to select, filter, and visualize quantitative and qualitative insights to validate the report's written results and conduct post-hoc analyses. This requires both interactive charts and data rendering for media data types, such as audio and video.

D4. **No-code authoring experience.** Model analysis is often done by multi-disciplinary teams with varying degrees of technical ability [54, 182]. Users should be able to write reports without deep technical expertise and without learning new and complex tooling.

D5. **Collaboration and publishing.** Reports are rarely authored by individuals and are often done by large teams. Multiple users should be able to collaborate on reports. Interactive reports should also be publishable and shareable directly from the authoring platform.

7.3 Zeno Reports

To fulfill these design requirements, we built a visual authoring tool for creating *Zeno Reports*, interactive AI evaluation reports. The authoring tool is a no-code platform that allows users to drag and drop different components into a notebook-style linear document (fig. 7.2). We considered whether reports should be in a dashboard- or notebook-style format but decided on a vertical notebook since that is the status quo for reports. The


Zeno GPT MT Benchmark Report 8

GPT MT Benchmark Report

Updated Saturday, October 28, 2023 at 4:17 PM

Author: First Author × Second Author ×

Linked Projects: GPT MT Benchmark ×

Why do LLMs lag?

Can we dig a bit deeper and explore why specifically LLMs perform worse? A common pattern we found browsing through the data was that GPT produced degenerated outputs containing the same word repeated over and over.

To quantify this behavior we looked at how many outputs each model had in which any word was repeated more than three times.

CHART

ChrF by Script (GPT MT Benchmark)

ChrF by Script
system
ChatGPT zero-shot
ChatGPT five-shot
GPT4 five-shot
NLLB MOE
Tamil Script
Arabic
Cyrillic
Georgian
Latin
Add **collaborators** & supporting **Zeno projects**
Interactively **reposition**, **edit**, & **delete** report cells
add cells
select & render **charts**, **slices**, & **tags** created in Zeno projects

Figure 7.2: The *Zeno Reports* authoring interface. Authors can use the UI-based editor to create, customize, and manage the components that make up a Zeno Report, including data slices and charts. Different cell types supported are detailed in Section section 7.3, including text, images, data visualizations, and interactive elements.

notebook structure also provides an author-driven linear narrative, following a martini glass narrative structure [59] that encourages the reader to read the report first and then explore.

Zeno Reports is built directly on the Zeno platform, which enables users to go directly from discovering and quantifying model behaviors to creating reports. The original Zeno platform had minimal reporting functionality, so users had to export their insights into other formats, such as PDF reports. The data and chart blocks in *Zeno Reports* are built directly from the insights and slices created using Zeno **(D2)**.

Three building block elements are available in *Zeno Reports*, detailed in the following sections. The first is *data rendering* blocks, which display specific inputs and the corresponding outputs for a model. The next are *chart* blocks, which quantify the model's behavior in interactive visualizations. Lastly, there are *markdown* blocks, which allow users to add narratives around the data and visualizations. Users can create and organize these blocks into a report using a visual drag-and-drop GUI **(D1, D3, D4)**.

Zeno Reports are deployed as a fully hosted platform with user accounts and organizations. All reports are collaborative so multiple users can add, edit, or remove content. Reports can also be published with a single click and shared publicly **(D5)**. The main reporting interface was built using Svelte and SvelteKit. The backend API was implemented using FastAPI to use Python-based AI libraries, with PostgreSQL as the database. Vega and VegaLite [185] were used to create the visualizations.

7.3.1 Data Rendering

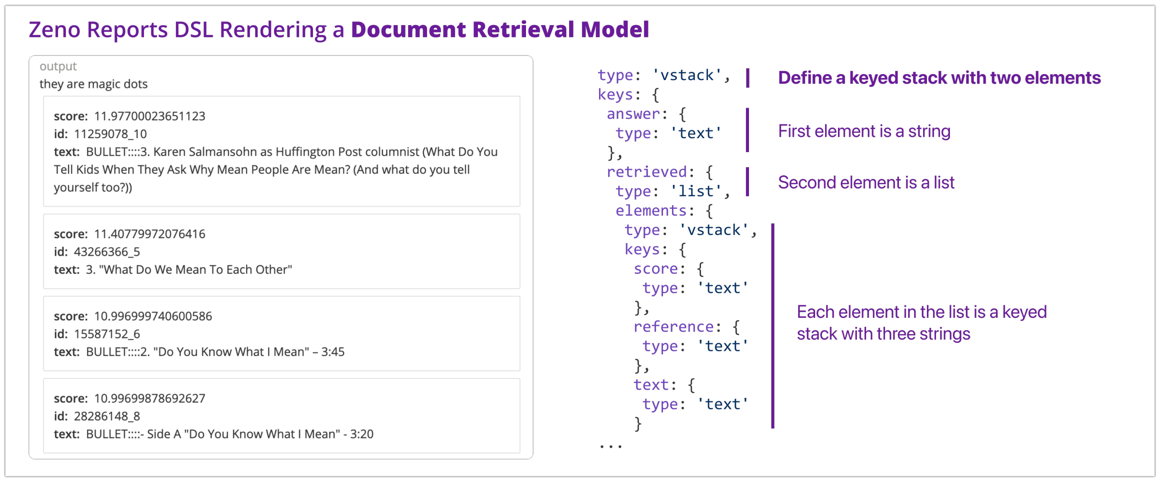
The first report element available for *Zeno Reports* is *data rendering* blocks for displaying AI inputs and outputs. When discussing model behavior, it is important to show examples of the model's output for specific input types. Looking at individual instances can inspire the reader to explore new analyses **(D3)**. It is also important to verify quantitative results empirically by seeing if the aggregate numbers match model outputs. In the subsequent results section, we will discuss an example of this happening in practice.

What makes data rendering challenging is the vast array of data that AI systems can input and/or output, from audio to images and video. While Zeno supports modular visualizations for different data, each visualization has to be implemented in React code and is not extensible to new data and output types. Additionally, Zeno does not support multimodal data inputs and/or outputs, which are becoming increasingly common.

For Zeno Reports, we designed a *domain-specific language* (DSL) that can specify how model inputs, labels, and outputs should be rendered. We considered a visual editor for specifying the data layout, but found that there were too many potential configurations and types of model inputs and outputs. A DSL is a middle ground that keeps the instance view extensible to diverse model types but reduces the technical knowledge and effort required to implement new views compared to Zeno's code-based renderer. The DSL has two types of elements: *display* elements that specify how to render raw data and *layout* components that specify how the display components should be organized. *Zeno Reports* currently support eight different *display* types (text, image, audio, code, markdown, message, separated values) and three *layout* types (list, vertical stack, table). To write a specification, a user defines the layout and display of a model's three data fields: the $input$, the ground truth $label$, and the model $output$.

An example specification can be seen in fig. 7.3. In this case, a user wants to render a list of output documents retrieved by a model and a textual answer derived from the documents. To visualize these data, they specified a set of nested vertical stacks and lists to obtain the output answer and a list of formatted documents.

The data rendering DSL is designed to be as expressive as possible, but is not easy to learn and use, as required by **D4**). *Zeno Reports* support passing in string identifiers for common DSL configurations, such as text generation or image classification. In deployment, we found that these views worked for most use cases or that one technical team member implemented the view so that other nontechnical users could use it.


Zeno Reports DSL Rendering a **Document Retrieval Model**
output
Define a keyed stack with two elements
First element is a string
Second element is a list
Each element in the list is a keyed stack with three strings

they are magic dots

score: 11.97700023651123
id: 11259078_10
text: BULLET::::3. Karen Salmansohn as Huffington Post columnist (What Do You Tell Kids When They Ask Why Mean People Are Mean? (And what do you tell yourself too?))

score: 11.40779972076416
id: 43266366_5
text: 3. "What Do We Mean To Each Other"

score: 10.996999740600586
id: 15587152_6
text: BULLET::::2. "Do You Know What I Mean" – 3:45

score: 10.99699878692627
id: 28286148_8
text: BULLET::::- Side A "Do You Know What I Mean" - 3:20

```
type: 'vstack',
keys: {
 answer: {
  type: 'text'
 },
 retrieved: {
  type: 'list',
  elements: {
   type: 'vstack',
   keys: {
    score: {
     type: 'text'
    },
    reference: {
     type: 'text'
    },
    text: {
     type: 'text'
    }
...
```

Figure 7.3: An example of Zeno Report's domain-specific language (DSL) for rendering AI data. In this example, a user has created a specification for rendering retrieved documents and a summarized answer. They combined vertical stack and list layouts to render different text fields.

7.3.2 Charts

In addition to showing direct data instances, reports should show *quantitative summaries* of model behavior **(D1)**. There is a rich history of GUI-based interfaces for creating charts, including research systems such as Voyager [186, 187] and Charticulator [188] and commercial systems such as Tableau [189]. We took inspiration from these systems to build a GUI-based chart builder for *Zeno Reports*.

Using Zeno's key insight that most AI evaluations can be reduced to a metric, slice, and model, we built a more focused chart-building interface with a limited set of encodings and chart types. Charts in *Zeno Reports* are limited to three data encodings (slices, metrics, and models), which can be assigned to three channels in the charts. These building blocks are tied directly to the corresponding primitives in Zeno, so updates to the underlying slices or metrics are reflected directly in the report charts **(D2)**.

The chart editing interface, seen in fig. 7.4, has a settings panel on the left-hand side and a chart preview on the right. In the settings panel, users can choose from six preset chart types **(7.4A)** and set the encoding for each chart axis **(7.4B)**. *Zeno Reports* currently supports six types of charts: bar charts, line charts, tables, beeswarm charts, radar charts, and heatmaps. We chose this subset by looking at report specifications such as model cards and the types of charts used in published evaluations. Users can select the visualization type they wish to display and then select what *variable* to encode across which *axis*. In the example shown in fig. 7.4, the user chose to show the performance of different models on slices of different languages for the translation task.

The charts are rendered using Vega and VegaLite [158]. This allows us to make the

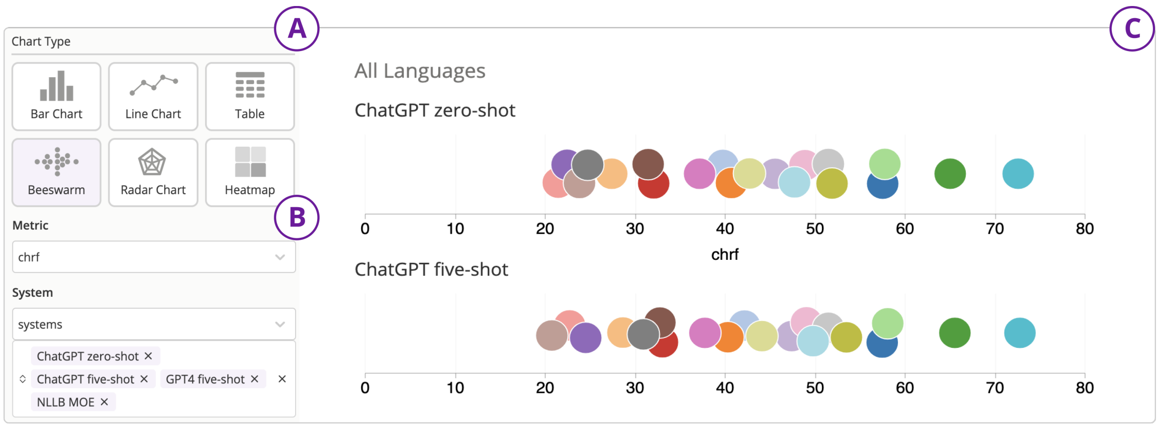


Figure 7.4: Interactive chart editor interface. (A) Select from six default chart types, from basic bar charts to advanced beeswarm and radar charts. (B) Define mappings from evaluation variables (slices, models, and metrics) to chart axes, facilitating customized data representation. (C) Live data previews on the right-hand side allow users to immediately see the impact of their configurations on the chart's appearance.

visualization interactive **(D3)** with tooltips and filtering. Additionally, new visualizations can be added with just a Vega specification, making it easy to add new visualizations for people who have experience with Vega.

7.3.3 Markdown

The last block type in *Zeno Reports* is the simple but important *text markdown* block. Users can write any standard markdown syntax, including images, code rendering, etc., rendered into formatted text. We specifically use GitHub's markdown syntax. The text block is essential for providing narrative around the data, a key aspect of *narrative visualization*.

7.4 Evaluation

We took two approaches to evaluate Zeno Reports. First, we released the hosted platform and core Zeno features to the general public. We also did a study with experts using Zeno Reports to analyze Google Gemini models, the first series of LLMs to claim parity with GPT 3.5 and 4 [190]. These evaluation approaches combined gave us enough data to do quantitative analysis on report creation patterns and qualitative insights into expert usage of the system.

7.4.1 Public Deployment

In September 2023, we deployed the hosted Zeno platform, including the core Zeno features and *Zeno Reports*, to the general public. Anyone could create an account, upload data and system outputs, and create reports. To encourage platform usage, we created integrations for common AI benchmarking tools such as the Eleuther Test Harness [191], HuggingFace OpenLLM Leaderboard [192], and RAGAS evaluation tool [193]. For the results of this study, we cut off data collection on February 20, 2024, giving us approximately five months' worth of data. We used anonymized statistical results to conduct quantitative analyses of the reports and understand the overall usage patterns of Zeno and Zeno Reports.

7.4.2 Gemini Benchmarking

For the second part of the evaluation, we wanted to understand how *experts* use *Zeno Reports* and obtain qualitative information about their reporting experience. To this end, we collaborated with a group of researchers conducting a deep evaluation of Google's transformer-based Gemini models [190]. The Gemini models were the first to claim parity with OpenAI's GPT 3.5 and GPT 4 models. The research group's study aimed to compare the Gemini Pro model, the publicly released model at the time, with GPT 3.5 and 4 across various capabilities. The team consisted of 9 researchers, including PhD students and professors, and lasted approximately two weeks.

To achieve this, the researchers ran each model on common benchmarks that measure capabilities such as common sense reasoning, mathematical logic, and translation ability. Each researcher in the project ran a subset of the benchmarks and generated output data for all models. These results were then uploaded to the hosted *Zeno Reports* platform, where the user performed an in-depth analysis, created charts, and wrote a final report. The reports were published with and linked to the final PDF analysis paper [183]. We collected their feedback through asynchronous Slack conversations and synchronous video calls.

7.5 Results

7.5.1 Quantitative Patterns of Report Authoring

During the five months in which the hosted version of Zeno and *Zeno Reports* were deployed, the platform had over 500 users sign up. Users created over 220 reports using over 2,000 slices and 1,700 report components. These reports evaluated over 15,000 AI models, and were read over 12,000 times. These numbers include the 11 reports created by the Gemini benchmarking team.

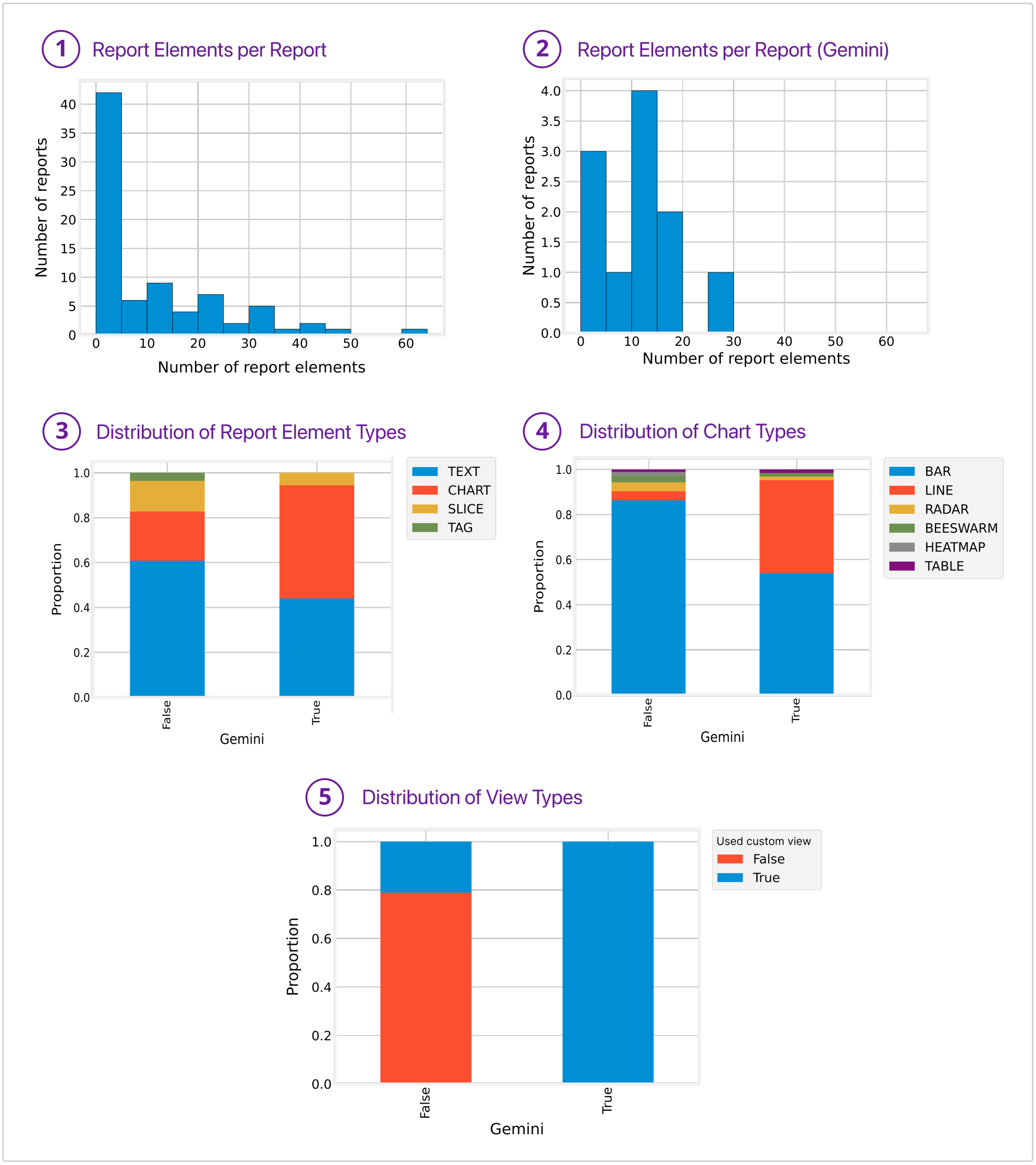


Figure 7.5: Quantitative results from the public deployment and expert usage of *Zeno Reports*. In charts 3-5, the left bar is for reports from the public while the right bar is for reports from the Gemini evaluation team. **(1 & 2)** Histograms showing the distribution of the number of report elements per report. **(3)** Distribution of report element types in reports. **(4)** Distribution of chart types in reports. **(5)** Distribution of view types in reports (string identifier vs. custom view with DSL).

We can first explore what the average Zeno Report looks like. Reports had an average of 11.75 cells and a median of 3 cells. We found that many reports only had one or two charts with a text cell, hence the large difference between the mean and median (fig. 7.5 1). On average, the reports were made up primarily of chart and markdown cells, with a much smaller proportion of slice cells and almost no tag cells (fig. 7.5 3). The Gemini analysis reports had a similar average of 11.36 cells and a median of 11 cells (fig. 7.5 2) - the higher median cell count is likely due to the lack of short test reports. Interestingly, the Gemini reports had a higher ratio of chart elements to text elements, with more than half of the elements being charts.

We can also explore the types of charts users put in their reports. Far and away, bar charts dominated reports, with over 80% of charts in public reports being bar charts (fig. 7.5 4). The Gemini researchers used significantly more line charts than the public, but neither population extensively used any of the more esoteric charts. We found that the line charts used by the Gemini team were often for fine-grained ablations of specific features, such as slicing data by the input length, which were less common in general reports.

Lastly, we can look at what data types models in reports used. About 80% of the public users used string-based view identifiers (e.g. text-classification) versus 20% who used custom DSL-based views. On the other hand, *all* of the Gemini reports used custom views. This was likely due to the specific format of their benchmarks, for example, custom string delimiters and setups. The *Zeno Reports* team also actively helped the Gemini team create new views for their tasks, lowering the barrier to entry.

7.5.2 Qualitative Insights on Using *Zeno Reports*

While working with the Gemini team, we gathered information about the reporting process of expert evaluators and the categorical advantages and limitations of *Zeno Reports*.

Using a combination of the base Zeno features and *Zeno Reports*, the auditors found significant limitations in the APIs of the audited models and the existing benchmarks. An example was the GSM-8K math reasoning task, a benchmark of grade-school math word problems [194]. The original benchmark implementation used a specific regular expression string to find the correct answer, but only counted it as correct if it was the last thing it said. The evaluator found by looking at instances that the output, in particular for advanced models, was often mid-way through the last sentence. Using this insight, they changed their regex to find the last number in the output and recalculated the scores. They found that *"using [the original] regex I get 67% accuracy and taking the last occurrence gets around 75%"*. They reported these findings in a report with the rest of the team using bar charts and slice cells. In another case, multiple auditors corroborated a common trend of the Gemini API to refuse to answer many types of questions. This impacted translation in particular, where Gemini refused to answer some questions if they were in a different language. The evaluator created multiple bar charts quantifying this pattern, which was shared with other team members who could reproduce the problem in their tasks.

We found that the tight coupling between the Zeno analysis platform and the reporting interface of *Zeno Reports* led to significant benefits. Toward the end of the evaluation process, the group decided to include the results for an additional model, Mixtral 8x7B [195]. To update their reports, users just had to push the results of the new model to Zeno, and all the charts and reports were dynamically updated with the new results. This saved significant work that would have been done to recreate and export the charts.

Despite what the evaluators were able to accomplish with *Zeno Reports*, a few feature limitations led them to supplement their analyses with other tools. Nearly all analyses the evaluators wanted to show, they were able to complete in Zeno. There were two exceptions, both of which were bar charts showing the *ratio* of different slices to each other. For example, in one task, a user wanted to show a stacked bar chart with the ratio of questions the model answered correctly, incorrectly, or refused to answer. This type of ratio-based visualization is not currently possible in the chart-building UI due to the GUI not implementing ratios as a possible encoding.

Another limitation was the amount of customization in the charts. Although the evaluators almost exclusively used bar and line charts, they wanted much more control over how they were rendered. For example, one user asked if *"There is any way to rescale the axis. For some tasks, the lines are clustered on the top of the line graph"*, while another wanted to *"show the entire x-axis label"* that had been truncated. *Zeno Reports* does not expose the controls for many fine-grained ways in which charts can be edited.

Lastly, the evaluators faced limitations with the interface between Zeno and existing reporting methods. The reporting team wanted to export the charts to use them in a LaTex-based PDF. Although *Zeno Reports* supports exporting charts, they require additional tweaking to render correctly in the PDF medium. One user mentioned that *"The current figure/font ratio makes the fonts too small when inserted in our paper"* and wanted additional controls to modify the rendering of the chart.

7.6 Discussion

Overall, we confirmed our finding in the Zeno case studies that the primitives of slices, metrics, and models were sufficient to represent almost every analysis users wanted to conduct. This included both the public users and expert evaluators conducting more fine-grained evaluations. The only exception we saw during deployment was for ratio-based charts, such as a stacked bar chart showing what slices make up what percentage of a dataset.

A surprising pattern we saw across users was the relative *simplicity* of building blocks used in reports. Almost all reports consisted primarily of text and chart elements, with one or two slice visualizations often shown at the start of a report as context for the task and dataset. Simplicity was also present in the types of charts users chose. While the expert evaluators used more line charts than the public, over 90% of the charts in both cohorts were just bar or line charts. Instead, we saw lots of nuance in how charts and reports were rendered, such as requests for tweaking chart rendering settings like font sizes and axes ranges.

We also repeatedly saw how sticky existing media is for sharing results. This was most apparent with the Gemini group, who needed to export the charts to embed into a PDF document. Many publication venues and administrative processes require standardized media such as PDF reports, which a system such as *Zeno Reports* needs to support.

Lastly, we confirmed the importance of tightly coupling the *analysis* and *reporting* phases of evaluating AI systems. Evaluators often went back and forth between their reports and analyses as they received feedback on their reports from collaborators and updated or augmented their analyses. One such example was when the Gemini evaluators added Mixtral 8x7B to their analyses and were able to update their reports quickly.

The overall trend in these results is that the complexity of AI evaluation is in the *analysis* stages, whereas the results should be summarized in a simple and easily digestible format. Finding and quantifying interesting model behaviors is complex, but the most poignant and relevant findings can almost always be summarized in a bar or line chart. Reports should do their best to ”get out of the way“ and let users share and update their analyses in the lowest-friction way possible.

7.7 Limitations and Future Work

Our evaluation was fully observational and did not directly measure the benefits of *Zeno Reports* over existing reporting methods such as Interactive Model Cards. In the future, a controlled laboratory study could measure how effective *Zeno Reports* is at some of our design goals, such as the speed at which reports can be updated and the extent to which readers can perform post-hoc analyses.

Despite our stated goal to remove the need to learn any new language or complex interface, many users had to interact with the domain-specific language to specify their instance views. The need for custom views, especially by the Gemini evaluation team, was much greater than we expected. Even within text-to-text tasks, many benchmarks had specific formatting requirements that needed a custom view. Novel abstractions may be needed to support the large space of potential data renderers with a relatively simple specification.

The charts and instances views in *Zeno Reports* are interactive but only support basic interactions. In some cases, empowering the user with more powerful interactions like cross-filtering charts or filtering for specific instances could be useful. In the framing of narrative visualization, this would make reports more reader-driven rather than fully author-driven. Future work could explore more powerful interactions with report elements.

7.8 Conclusion

Understanding and reporting the behavior of AI systems is growing in importance as they become more powerful and exhibit more complex behaviors. AI analysis is a time-intensive, multi-stage process that includes everything from discovering and quantifying behaviors to creating visualizations. In this work, we introduced *Zeno Reports*, a platform that makes creating interactive, reproducible reports of AI behavior easier. We hope this work inspires the development of analysis tools that push us toward a future of performant and equitable AI systems.

CHAPTER 8
CONCLUSIONS

8.1 Discussion

This thesis encompasses five years of work on helping people understand the behaviors of AI systems. In general, my hypothesis that behaviors are a core abstraction for understanding and developing AI was validated. Using the behavior-based sensemaking framework, I built systems that helped users, from AI developers to end users, accomplish tasks including debugging, auditing, and human-AI collaboration. I also built a general-purpose analysis and reporting tool, Zeno, used by over 500 users in myriad domains.

8.1.1 Practitioners spend most of their time discovering behaviors

While I found that a sensemaking framework adequately represented the AI analysis process, I also found that the relative importance of the steps is quite unbalanced. If we split the analysis steps into two phases, *"discovery"* (instances and schemas) and *"evaluation"* (hypotheses and reports), practitioners primarily find the *"discovery"* phase to be the most critical and challenging. I found that practitioners focused on finding evidence of a potential behavior and often did not need to rigorously validate or formalize their findings in a report.

Discovery is even more critical for modern AI systems built on large foundation models. With classic supervised learning models, developers require a large dataset and labels to train a model. In contrast, with foundation models, a user can specify an entire AI system with complex behavior using a single paragraph of text. For these models, AI developers don't have any instances or labels to do analysis. Thus, they focus on the first stage of collecting and labeling data that can surface interesting behaviors rather than creating schemas and formal hypotheses around existing data.

8.1.2 Collecting or generating instances is becoming a challenging sensemaking step

The importance of collecting or generating input instances was a critical insight that I did not address directly in my thesis work and is becoming more central with the proliferation of data-free LLM-based models. Some of my earlier work that did not make it into this thesis explored the instance gathering problem with "failure reports", crowdsourced descriptions of AI errors [19]. However, the Zeno and Zeno Reports platforms were built assuming that users have an existing labeled dataset to analyze. This design decision reflected the pre-foundation model era, where most AI systems were supervised learning models trained on existing datasets. When we publicly deployed Zeno in the post-foundation model era, we spoke with numerous users who wanted to use Zeno for in-depth analysis but did not have any data to ingest into the system.

This limitation is a massive opportunity for future AI analysis tools. There is already a growing body of work on generating synthetic data to discover model limitations, such as the AdaTest [50] tool, which uses human guidance to generate challenging new test cases. While prompt-based models are less likely to have evaluation data, foundation models also enable the generation of realistic synthetic data that can be used to identify interesting behaviors.

8.1.3 Limited need for formally verifying behaviors

In the other half of the sensemaking process, I was surprised by the relative lack of importance many practitioners place on formally quantifying behaviors. The early work in this thesis was inspired by studies quantifying disparities in classification models, such as the now-famous Gender Shades study that found significant disparities in gender classification models [13]. These studies focused on tasks with relatively simple outputs and well-defined behaviors with clear societal impact. Newer models are applied to domains beyond classification and regression with more complex and nuanced behaviors. A 10% disparity in loan approval rates between demographic groups has a clear and significant real-world impact. An image captioning model may have the interesting behavior that it cannot accurately count the number of people in an image, but it is not clear that formally calculating how often this happens is necessary. It is also often the case that seeing a few example failures is enough to inform a model update or deployment decision without knowing statistically how often it happens. This was something that was quite different between academic and industry practitioners - academic researchers placed a much higher value on formal, quantified analyses.

I was also surprised by the relatively lackluster demand and use of formal reporting methods. We built Zeno Reports informed by the increasing usage of reports in practice and empirical studies on the challenges in the authoring and updating of reports. In practice, we found that the type of reporting practitioners do has a bimodal distribution. On one side, formal reporting methods like model cards are often integrated into existing platforms for hosting and sharing models. There is no incentive for users to use a separate reporting platform to author their reports when they have to re-write them in a specific format and template for a specific platform. On the other side of the spectrum are AI development teams who share quick insights between themselves as they build an AI system. For this type of "reporting", simply taking a screenshot of a chart or an example failure and sending it with some context in an email or message was generally sufficient. Zeno Reports were most valuable for a blend of these reporting needs - they have more structure and information than single charts but are not directly tied to downstream requirements or platforms. Future work on reporting may benefit from a deep understanding of existing reporting pathways, such as policy requirements, and the development of tools specifically for those types of reports.

8.1.4 Impact of incentives on tool use and adoption

Lastly, we found that incentive alignment, cultural dynamics, and policy played an essential role in the use and adoption of analysis tools, as found by existing empirical work cited in this thesis [53, 182]. One prototypical example is the goals and skillsets of engineering versus product and management roles. Engineers are incentivized to build better performing systems that match the specifications and requirements given to them. They have the technical breadth to run deep analyses and identify limitations that can help them improve an AI system. Product and management roles might be required to submit reports to regulators or mitigate potential PR crises from specific model behaviors. Unlike engineering teams, they often do not have the skill set or familiarity with the underlying AI system to run the required analyses and must convince engineering to collaborate with them. The culture of organizations that build AI-powered products will likely need to change to encourage the use of specific analysis tools. For example, engineering teams may have to be incentivized to use analysis tools like Zeno that non-technical stakeholders can extend for post-hoc analyses.

8.2 Conclusion

The rise of models that can be guided with text prompts or small datasets has empowered millions to create complex AI systems, from chatbots to creative writing partners. As the tasks AI systems tackle increase in complexity, it becomes more challenging to understand and improve model behavior in line with how a practitioner expects a model to perform. Instead of centering AI development on the underlying technology enabling it, such as the training data and model architecture, centering model iteration on the expected behaviors can lead to more fair, safe, and performant models.

This thesis introduced Behavior-Driven AI Development (BDAI), a new philosophy that centers the development of models on their *behavior* across diverse inputs. The thesis first sets the foundation for BDAI by interviewing dozens of AI practitioners to understand how they approach understanding and iterating on their models. It then formally defines AI behavior and how practitioners make sense of behavior. Additionally, I show how these insights can be re-used to improve the accuracy of domain experts collaborating with AI assistants by calibrating their reliance. I then designed and implemented Zeno and Zeno Reports, a general-purpose platform for defining and tracking AI behaviors that acts as the hub for BDAI.

The BDAI philosophy focuses AI development centrally on what practitioners and users care about - the direct output of a model. By formally defining a development paradigm and accompanying tools, I hope to encourage future work that explores how AI development is done and future tools for creating responsible AI systems.

REFERENCES

[1] M. Bojarski *et al.*, “End to End Learning for Self-Driving Cars,” pp. 1–9, 2016.

[2] B. Zhang, P. Liang, X. Zhou, A. Ahmad, and M. Waseem, “Practices and Challenges of Using GitHub Copilot: An Empirical Study,” arXiv:2303.08733 [cs], Jul. 2023, pp. 124–129.

[3] W. Bulten *et al.*, “Artificial intelligence assistance significantly improves Gleason grading of prostate biopsies by pathologists,” *Modern Pathology*, vol. 34, no. 3, pp. 660–671, Mar. 2021.

[4] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs], Dec. 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[6] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, ISSN: 10495258, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[7] G. Bansal, B. Nushi, E. Kamar, W. S. Lasecki, D. S. Weld, and E. Horvitz, “Beyond Accuracy: The Role of Mental Models in Human-AI Team Performance,” *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 7, no. 1, p. 19, 2019.

[8] D. A. Norman, “Some Observations on Mental Models,” in *Mental Models*, D. Gentner and A. L. Stevens, Eds., 0th ed., Psychology Press, Jan. 2014, pp. 15–22, ISBN: 978-1-315-80272-5.

[9] M. Haenlein and A. Kaplan, “A brief history of artificial intelligence: On the past, present, and future of artificial intelligence,” *California management review*, vol. 61, no. 4, pp. 5–14, 2019.

[10] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] R. Bommasani *et al.*, *On the Opportunities and Risks of Foundation Models*, arXiv:2108.07258 [cs], Jul. 2022.

[12] S. Amershi *et al.*, "Software Engineering for Machine Learning: A Case Study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, QC, Canada: IEEE, May 2019, pp. 291–300, ISBN: 978-1-72811-760-7.

[13] J. Buolamwini and T. Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, S. A. Friedler and C. Wilson, Eds., ser. Proceedings of Machine Learning Research, Buolamwini2018, vol. 81, PMLR, Feb. 2018, pp. 77–91.

[14] I. Rahwan *et al.*, "Machine Behaviour," *Nature*, vol. 568, no. 7753, pp. 477–486, Apr. 2019.

[15] A. Cabrera, M. Tulio Ribeiro, B. Lee, R. Deline, A. Perer, and S. M. Drucker, "What Did My AI Learn? How Data Scientists Make Sense of Model Behavior," *ACM Trans. Comput.-Hum. Interact.*, vol. 30, no. 1, Mar. 2023, Place: New York, NY, USA Publisher: Association for Computing Machinery.

[16] C. Yang, R. Brower-Sinning, G. A. Lewis, C. Kästner, and T. Wu, *Capabilities for Better ML Engineering*, arXiv:2211.06409 [cs], Nov. 2022.

[17] Z. Pei, L. Liu, C. Wang, and J. Wang, "Requirements Engineering for Machine Learning: A Review and Reflection," in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, Melbourne, Australia: IEEE, Aug. 2022, pp. 166–175, ISBN: 978-1-66546-000-2.

[18] H. Subramonyam, C. Seifert, and E. Adar, "Towards A Process Model for Co-Creating AI Experiences," in *DIS 2021 - Proceedings of the 2021 ACM Designing Interactive Systems Conference: Nowhere and Everywhere*, Association for Computing Machinery, Inc, Jun. 2021, pp. 1529–1543, ISBN: 978-1-4503-8476-6.

[19] A. Cabrera, A. J. Druck, J. I. Hong, and A. Perer, "Discovering and Validating AI Errors With Crowdsourced Failure Reports," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW2, pp. 1–22, Oct. 2021.

[20] K. E. Weick, *Sensemaking in Organizations* (Foundations for organizational science). Thousand Oaks: Sage Publications, 1995, ISBN: 978-0-8039-7176-9 978-0-8039-7177-6.

[21] K. E. Weick, K. M. Sutcliffe, and D. Obstfeld, "Organizing and the Process of Sensemaking," *Organization Science*, vol. 16, no. 4, pp. 409–421, Aug. 2005.

[22] S. Maitlis and M. Christianson, "Sensemaking in Organizations: Taking Stock and Moving Forward," *Academy of Management Annals*, vol. 8, no. 1, pp. 57–125, Jan. 2014.

[23] D. Ancona, "Sensemaking: Framing and Acting in the Unknown," *The Handbook for Teaching Leadership: Knowing, doing, and being*, vol. 10, no. 3, pp. 3–19, 2012.

[24] D. Ucbasaran, D. A. Shepherd, A. Lockett, and S. J. Lyon, "Life After Business Failure: The Process and Consequences of Business Failure for Entrepreneurs," *Journal of Management*, vol. 39, no. 1, pp. 163–202, Jan. 2013.

[25] H. Kaur, E. Adar, E. Gilbert, and C. Lampe, "Sensible AI: Re-imagining Interpretability and Explainability using Sensemaking Theory," *arXiv:2205.05057 [cs]*, May 2022, arXiv: 2205.05057.

[26] G. Whiteman and W. H. Cooper, "Ecological Sensemaking," *Academy of Management Journal*, vol. 54, no. 5, pp. 889–911, Oct. 2011.

[27] J. C. Chang, N. Hahn, and A. Kittur, "Mesh: Scaffolding Comparison Tables for Online Decision Making," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, Virtual Event USA: ACM, Oct. 2020, pp. 391–405, ISBN: 978-1-4503-7514-6.

[28] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card, "The cost structure of sensemaking," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '93*, Amsterdam, The Netherlands: ACM Press, 1993, pp. 269–276, ISBN: 978-0-89791-575-5.

[29] P. Pirolli and S. Card, "The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis," *Proceedings of International Conference on Intelligence Analysis*, vol. 2005, no. January, pp. 2–4, 2005.

[30] V. Grigoreanu, M. Burnett, S. Wiedenbeck, J. Cao, K. Rector, and I. Kwan, "End-user Debugging Strategies: A Sensemaking Perspective," *ACM Transactions on Computer-Human Interaction*, vol. 19, no. 1, pp. 1–28, Mar. 2012.

[31] D. H. Chau, A. Kittur, J. I. Hong, and C. Faloutsos, "Apolo: Making Sense of Large Network Data by Combining Rich User Interaction and Machine Learning," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vancouver BC Canada: ACM, May 2011, pp. 167–176, ISBN: 978-1-4503-0228-9.

[32] H. Bosch *et al.*, “ScatterBlogs2: Real-Time Monitoring of Microblog Messages through User-Guided Filtering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2022–2031, Dec. 2013.

[33] T. Kulesza, S. Stumpf, M. Burnett, and I. Kwan, “Tell me more?: The effects of mental model soundness on personalizing an intelligent agent,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Austin Texas USA: ACM, May 2012, pp. 1–10, ISBN: 978-1-4503-1015-4.

[34] J. D. Lee and K. A. See, “Trust in Automation: Designing for Appropriate Reliance,” *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 46, no. 1, pp. 50–80, Jan. 2004.

[35] R. Tomsett *et al.*, “Rapid Trust Calibration through Interpretable and Uncertainty-Aware AI,” *Patterns*, vol. 1, no. 4, p. 100 049, 2020, Publisher: Elsevier Inc.

[36] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W. K. Wong, “Too Much, Too Little, or Just Right? Ways Explanations Impact End Users’ Mental Models,” *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, pp. 3–10, 2013.

[37] R. Kocielnik, S. Amershi, and P. N. Bennett, “Will You Accept an Imperfect AI?: Exploring Designs for Adjusting End-user Expectations of AI Systems,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk: ACM, May 2019, pp. 1–14, ISBN: 978-1-4503-5970-2.

[38] Y. Zhang, Q. V. Liao, and R. K. E. Bellamy, “Effect of confidence and explanation on accuracy and trust calibration in AI-assisted decision making,” in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, Barcelona Spain: ACM, Jan. 2020, pp. 295–305, ISBN: 978-1-4503-6936-7.

[39] P. Khadpe, R. Krishna, L. Fei-Fei, J. T. Hancock, and M. S. Bernstein, “Conceptual Metaphors Impact Perceptions of Human-AI Collaboration,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW2, 2020.

[40] V. Lai, H. Liu, and C. Tan, “”Why is ’Chicago’ Deceptive?” Towards Building Model-Driven Tutorials for Humans,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, 2020, pp. 1–13, ISBN: 978-1-4503-6708-0.

[41] J. Martinez, K. Gal, E. Kamar, and L. H. S. Lelis, “Personalization in Human-AI Teams: Improving the Compatibility-Accuracy Tradeoff,” *arXiv:2004.02289 [cs]*, Aug. 2020.

[42] K. Okamura and S. Yamada, "Adaptive trust calibration for human-AI collaboration," *PLoS ONE*, vol. 15, no. 2, pp. 1–20, 2020.

[43] H. Mozannar, A. Satyanarayan, and D. Sontag, "Teaching Humans When To Defer to a Classifier via Exemplars," *arXiv:2111.11297 [cs]*, Dec. 2021, arXiv: 2111.11297.

[44] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond Accuracy: Behavioral Testing of {NLP} Models with {C}heck{L}ist," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 4902–4912.

[45] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh, "ModelTracker: Redesigning Performance Analysis Tools for Machine Learning," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, Seoul Republic of Korea: ACM, Apr. 2015, pp. 337–346, ISBN: 978-1-4503-3145-6.

[46] T. Wu, M. T. Ribeiro, J. Heer, and D. Weld, "{E}rrudite: Scalable, Reproducible, and Testable Error Analysis," *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pp. 747–763, 2019.

[47] A. Cabrera, W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau, "FairVis: Visual Analytics for Discovering Intersectional Bias in Machine Learning," in *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2019, pp. 46–56, ISBN: 978-1-72812-284-7.

[48] Y. Chung, T. Kraska, N. Polyzotis, K. H. Tae, and S. E. Whang, "Slice Finder: Automated Data Slicing for Model Validation," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, Macao, Macao: IEEE, Apr. 2019, pp. 1550–1553, ISBN: 978-1-5386-7474-1.

[49] S. Eyuboglu *et al.*, "Domino: Discovering Systematic Errors with Cross-Modal Embeddings," *arXiv:2203.14960 [cs]*, Apr. 2022, arXiv: 2203.14960.

[50] M. T. Ribeiro and S. Lundberg, "Adaptive Testing and Debugging of NLP Models," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 3253–3267.

[51] M. Mitchell *et al.*, "Model Cards for Model Reporting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Atlanta GA USA: ACM, Jan. 2019, pp. 220–229, ISBN: 978-1-4503-6125-5.

[52] M. Arnold *et al.*, “FactSheets: Increasing trust in AI services through supplier’s declarations of conformity,” *IBM Journal of Research and Development*, vol. 63, no. 4/5, 6:1–6:13, Jul. 2019.

[53] N. Nahar, S. Zhou, G. Lewis, and C. Kästner, “Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process,” *arXiv:2110.10234 [cs]*, Dec. 2021, arXiv: 2110.10234.

[54] A. Bäuerle *et al.*, “Symphony: Composing Interactive Interfaces for Machine Learning,” in *CHI Conference on Human Factors in Computing Systems*, New Orleans LA USA: ACM, Apr. 2022, pp. 1–14, ISBN: 978-1-4503-9157-3.

[55] J. Françoise, B. Caramiaux, and T. Sanchez, “Marcelle: Composing Interactive Machine Learning Workflows and Interfaces,” in *The 34th Annual ACM Symposium on User Interface Software and Technology*, Virtual Event USA: ACM, Oct. 2021, pp. 39–53, ISBN: 978-1-4503-8635-7.

[56] T. Gebru *et al.*, “Datasheets for datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, Dec. 2021.

[57] J. Stoyanovich and B. Howe, “Nutritional Labels for Data and Models,” *IEEE Data Eng. Bull.*, vol. 42, pp. 13–23, 2019.

[58] A. Crisan, M. Drouhard, J. Vig, and N. Rajani, “Interactive Model Cards: A Human-Centered Approach to Model Documentation,” in *2022 ACM Conference on Fairness, Accountability, and Transparency*, Seoul Republic of Korea: ACM, Jun. 2022, pp. 427–439, ISBN: 978-1-4503-9352-2.

[59] E Segel and J Heer, “Narrative Visualization: Telling Stories with Data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1139–1148, Nov. 2010.

[60] M. Conlen and J. Heer, “Idyll: A Markup Language for Authoring and Publishing Interactive Articles on the Web,” in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, Berlin Germany: ACM, Oct. 2018, pp. 977–989, ISBN: 978-1-4503-5948-1.

[61] M. Conlen, M. Vo, A. Tan, and J. Heer, “Idyll Studio: A Structured Editor for Authoring Interactive & Data-Driven Articles,” in *The 34th Annual ACM Symposium on User Interface Software and Technology*, Virtual Event USA: ACM, Oct. 2021, pp. 1–12, ISBN: 978-1-4503-8635-7.

[62] A. Cabrera *et al.*, “Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning,” in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’23, event-place: Hamburg, Germany, New York,

NY, USA: Association for Computing Machinery, 2023, ISBN: 978-1-4503-9421-5.

[63] A. Hopkins and S. Booth, "Machine Learning Practices Outside Big Tech: How Resource Constraints Challenge Responsible Development," in *AIES 2021 - Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, Association for Computing Machinery, Inc, Jul. 2021, pp. 134–145, ISBN: 978-1-4503-8473-5.

[64] L. Chen, T. Cai, M. Zaharia, and J. Zou, "Did the Model Change? Efficiently Assessing Machine Learning API Shifts," *arXiv:2107.14203 [cs, stat]*, Jul. 2021, arXiv: 2107.14203.

[65] E. Foong, D. Gergle, and E. M. Gerber, "Novice and Expert Sensemaking of Crowdsourced Design Feedback," *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. CSCW, pp. 1–18, Dec. 2017.

[66] T. D. Wang, K. Wongsuphasawat, C. Plaisant, and B. Shneiderman, "Extracting Insights from Electronic Health Records: Case Studies, a Visual Analytics Process Model, and Design Recommendations," *Journal of Medical Systems*, vol. 35, no. 5, pp. 1135–1152, Oct. 2011.

[67] K. Holstein, J. Wortman Vaughan, H. Daumé, M. Dudik, and H. Wallach, "Improving Fairness in Machine Learning Systems: What Do Industry Practitioners Need?" In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk: ACM, May 2019, pp. 1–16, ISBN: 978-1-4503-5970-2.

[68] L. Oakden-Rayner, J. Dunnmon, G. Carneiro, and C. Re, "Hidden stratification causes clinically meaningful failures in machine learning for medical imaging," in *Proceedings of the ACM Conference on Health, Inference, and Learning*, Toronto Ontario Canada: ACM, Apr. 2020, pp. 151–159, ISBN: 978-1-4503-7046-2.

[69] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "Data Scientists in Software Teams: State of the Art and Challenges," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1024–1038, Nov. 2018.

[70] J. Attenberg, P. G. Ipeirotis, and F. Provost, "Beat the Machine: Challenging Workers to Find the Unknown Unknowns," 2011.

[71] M. Kahng, D. Fang, and D. H. P. Chau, "Visual Exploration of Machine Learning Results Using Data Cube Analysis," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics - HILDA '16*, San Francisco, California: ACM Press, 2016, pp. 1–6, ISBN: 978-1-4503-4207-0.

[72] M. Beaudouin-Lafon, “Designing Interaction, not Interfaces,” in *Proceedings of the working conference on Advanced visual interfaces - AVI ’04*, Gallipoli, Italy: ACM Press, 2004, p. 15, ISBN: 978-1-58113-867-2.

[73] S. Timmermans and I. Tavory, “Theory Construction in Qualitative Research: From Grounded Theory to Abductive Analysis,” *Sociological Theory*, vol. 30, no. 3, pp. 167–186, Sep. 2012.

[74] A. L. Strauss and J. M. Corbin, Eds., *Grounded theory in practice*. Thousand Oaks: Sage Publications, 1997, ISBN: 978-0-7619-0747-3 978-0-7619-0748-0.

[75] D. Kiela *et al.*, “Dynabench: Rethinking Benchmarking in NLP,” *arXiv:2104.14337 [cs]*, Apr. 2021, arXiv: 2104.14337.

[76] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–11, 2015.

[77] T. Karras, S. Laine, and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 4396–4405, ISBN: 978-1-72813-293-8.

[78] C. Shorten and T. M. Khoshgoftaar, “A Survey on Image Data Augmentation for Deep Learning,” *Journal of Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.

[79] D. Cashman *et al.*, “CAVA: A Visual Analytics System for Exploratory Columnar Data Augmentation Using Knowledge Graphs,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1731–1741, Feb. 2021.

[80] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: Rapid training data creation with weak supervision,” *Proceedings of the VLDB Endowment*, vol. 11, no. 3, pp. 269–282, Nov. 2017.

[81] A. Liu, S. Guerra, I. Fung, G. Matute, E. Kamar, and W. Lasecki, “Towards Hybrid Human-AI Workflows for Unknown Unknown Detection,” in *Proceedings of The Web Conference 2020*, Taipei Taiwan: ACM, Apr. 2020, pp. 2432–2442, ISBN: 978-1-4503-7023-3.

[82] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, and J. Wilson, “The What-If Tool: Interactive Probing of Machine Learning Models,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019.

[83] B. Nushi, E. Kamar, and E. Horvitz, “Towards Accountable AI: Hybrid Human-Machine Analyses for Characterizing System Failure,” in *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 6, 2018, p. 10.

[84] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz, “Identifying Unknown Unknowns in the Open World: Representations and Policies for Guided Exploration,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI’17, AAAI Press, 2017, pp. 2124–2132.

[85] G. d’Eon, J. d’Eon, J. R. Wright, and K. Leyton-Brown, “The Spotlight: A General Method for Discovering Systematic Errors in Deep Learning Models,” *arXiv:2107.00758 [cs, stat]*, Oct. 2021, arXiv: 2107.00758.

[86] S. Singla, B. Nushi, S. Shah, E. Kamar, and E. Horvitz, “Understanding Failures of Deep Networks via Robust Feature Extraction,” *arXiv:2012.01750 [cs]*, Jun. 2021, arXiv: 2012.01750.

[87] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, “Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 61–70, Jan. 2017.

[88] M. Pushkarna, J. Wexler, and J. Wilson, *Facets: An Open Source Visualization Tool for Machine Learning Training Data*, 2017.

[89] N.-C. Chen, J. Suh, J. Verwey, G. Ramos, S. Drucker, and P. Simard, “AnchorViz: Facilitating Classifier Error Discovery through Interactive Semantic Data Exploration,” in *23rd International Conference on Intelligent User Interfaces*, Tokyo Japan: ACM, Mar. 2018, pp. 269–280, ISBN: 978-1-4503-4945-1.

[90] H. Wang, X. Wu, Z. Huang, and E. P. Xing, “High-Frequency Component Helps Explain the Generalization of Convolutional Neural Networks,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 8681–8691, ISBN: 978-1-72817-168-5.

[91] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, “DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems,” in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Montpellier France: ACM, Sep. 2018, pp. 132–142, ISBN: 978-1-4503-5937-5.

[92] Y. Tian, K. Pei, S. Jana, and B. Ray, “DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars,” in *Proceedings of the 40th International Conference on Software Engineering*, Gothenburg Sweden: ACM, May 2018, pp. 303–314, ISBN: 978-1-4503-5638-1.

[93] P. He, C. Meister, and Z. Su, “Structure-Invariant Testing for Machine Translation,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, Seoul South Korea: ACM, Jun. 2020, pp. 961–973, ISBN: 978-1-4503-7121-6.

[94] K. Goel *et al.*, “Robustness Gym: Unifying the NLP Evaluation Landscape,” pp. 1–34, 2021.

[95] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, “How does Machine Learning Change Software Development Practices?” *IEEE Transactions on Software Engineering*, pp. 1–1, 2020.

[96] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo, "“Everyone wants to do the model work, not the data work”: Data Cascades in High-Stakes AI,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, Yokohama Japan: ACM, May 2021, pp. 1–15, ISBN: 978-1-4503-8096-6.

[97] S. G. Harris and S. G. Harris, “Organizational Culture and Individual Sensemaking : A Schema-based Perspective,” *INFORMS*, vol. 5, no. 3, pp. 309–321, 1994.

[98] D. Piorkowski, S. Park, A. Y. Wang, D. Wang, M. Muller, and F. Portnoy, “How AI Developers Overcome Communication Challenges in a Multidisciplinary Team: A Case Study,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–25, Apr. 2021.

[99] H. Shen, H. Jin, A. Cabrera, A. Perer, H. Zhu, and J. I. Hong, “Designing Alternative Representations of Confusion Matrices to Support Non-Expert Public Understanding of Algorithm Performance,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW2, pp. 1–22, Oct. 2020.

[100] Q. Yang, J. Suh, N.-C. Chen, and G. Ramos, “Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models,” in *Proceedings of the 2018 Designing Interactive Systems Conference*, Hong Kong China: ACM, Jun. 2018, pp. 573–584, ISBN: 978-1-4503-5198-0.

[101] J. Görtler *et al.*, “Neo: Generalizing Confusion Matrix Visualization to Hierarchical and Multi-Output Labels,” in *CHI Conference on Human Factors in Computing Systems*, New Orleans LA USA: ACM, Apr. 2022, pp. 1–13, ISBN: 978-1-4503-9157-3.

[102] A. Hinterreiter *et al.*, “ConfusionFlow: A model-agnostic visualization for temporal analysis of classifier confusion,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2020.

[103] G. Bansal and D. S. Weld, “A Coverage-Based Utility Model for Identifying Unknown Unknowns,” in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018, pp. 1463–1470.

[104] X. Zhang, Y. Yang, Y. Feng, and Z. Chen, “Software Engineering Practice in the Development of Deep Learning Applications,” *arXiv:1910.03156 [cs]*, Oct. 2019.

[105] S. R. Hong, J. Hullman, and E. Bertini, “Human Factors in Model Interpretability: Industry Practices, Challenges, and Needs,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW1, pp. 1–26, May 2020.

[106] M. K. Hong, A. Fourney, D. DeBellis, and S. Amershi, “Planning for Natural Language Failures with the AI Playbook,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, Yokohama Japan: ACM, May 2021, pp. 1–11, ISBN: 978-1-4503-8096-6.

[107] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, “Machine Learning Testing: Survey, Landscapes and Horizons,” *IEEE Transactions on Software Engineering*, pp. 1–1, 2020.

[108] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, “Model assertions for monitoring and improving ML models,” *Proceedings of Machine Learning and Systems*, vol. 2, pp. 481–496, 2020.

[109] M. A. Madaio, L. Stark, J. Wortman Vaughan, and H. Wallach, “Co-Designing Checklists to Understand Organizational Challenges and Opportunities around Fairness in AI,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–14, ISBN: 978-1-4503-6708-0.

[110] H. Shen, “Interactive notebooks: Sharing the code,” *Nature*, vol. 515, no. 7525, pp. 151–152, Nov. 2014.

[111] C. K. Ch’ng and C. S. Chan, “Total-Text: A Comprehensive Dataset for Scene Text Detection and Recognition,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto: IEEE, Nov. 2017, pp. 935–942, ISBN: 978-1-5386-3586-5.

[112] C. Callison-Burch, M. Osborne, and P. Koehn, “Re-evaluating the Role of Bleu in Machine Translation Research,” in *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy: Association for Computational Linguistics, Apr. 2006, pp. 249–256.

[113] G. Harboe and E. M. Huang, “Real-World Affinity Diagramming Practices: Bridging the Paper-Digital Gap,” in *Proceedings of the 33rd Annual ACM Conference on*

Human Factors in Computing Systems, Seoul Republic of Korea: ACM, Apr. 2015, pp. 95–104, ISBN: 978-1-4503-3145-6.

[114] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich, “Data Mountain: Using Spatial Memory for Document Management,” in *Proceedings of the 11th annual ACM symposium on User interface software and technology - UIST ’98*, San Francisco, California, United States: ACM Press, 1998, pp. 153–162, ISBN: 978-1-58113-034-8.

[115] F. Lekschas, X. Zhou, W. Chen, N. Gehlenborg, B. Bach, and H. Pfister, “A Generic Framework and Library for Exploration of Small Multiples through Interactive Piling,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 358–368, Feb. 2021.

[116] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778, ISBN: 978-1-4673-8851-1.

[117] C. Low, E. McCamey, C. Gleason, P. Carrington, J. P. Bigham, and A. Pavel, “Twitter A11y: A Browser Extension to Describe Images,” in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, Pittsburgh PA USA: ACM, Oct. 2019, pp. 551–553, ISBN: 978-1-4503-6676-2.

[118] T. Li, K. Luther, and C. North, “CrowdIA: Solving Mysteries with Crowdsourced Sensemaking,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–29, Nov. 2018.

[119] Y. Rogers, *HCI Theory*. 2012, ISBN: 978-1-60845-900-1.

[120] K. Fisher, S. Counts, and A. Kittur, “Distributed Sensemaking: Improving Sensemaking by Leveraging the Efforts of Previous Users,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Austin Texas USA: ACM, May 2012, pp. 247–256, ISBN: 978-1-4503-1015-4.

[121] A. Kittur, A. M. Peters, A. Diriye, and M. Bove, “Standing on the schemas of giants: Socially augmented information foraging,” in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, Baltimore Maryland USA: ACM, Feb. 2014, pp. 999–1010, ISBN: 978-1-4503-2540-0.

[122] C. J. Cai, S. Winter, D. Steiner, L. Wilcox, and M. Terry, “”Hello AI”: Uncovering the Onboarding Needs of Medical Practitioners for Human-AI Collaborative Decision-Making,” *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, Nov. 2019.

[123] M. De-Arteaga, R. Fogliato, and A. Chouldechova, “A Case for Humans-in-the-Loop: Decisions in the Presence of Erroneous Algorithmic Scores,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–12, ISBN: 978-1-4503-6708-0.

[124] A. Kawakami *et al.*, “Improving Human-AI Partnerships in Child Welfare: Understanding Worker Practices, Challenges, and Desires for Algorithmic Decision Support,” in *CHI Conference on Human Factors in Computing Systems*, New Orleans LA USA: ACM, Apr. 2022, pp. 1–18, ISBN: 978-1-4503-9157-3.

[125] Z. Buçinca, M. B. Malaya, and K. Z. Gajos, “To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-assisted Decision-making,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–21, Apr. 2021.

[126] M. Jacobs, M. F. Pradier, T. H. McCoy, R. H. Perlis, F. Doshi-Velez, and K. Z. Gajos, “How machine-learning recommendations influence clinician treatment selections: The example of antidepressant selection,” *Translational Psychiatry*, vol. 11, no. 1, p. 108, Jun. 2021.

[127] A. Bussone, S. Stumpf, and D. O’Sullivan, “The role of explanations on trust and reliance in clinical decision support systems,” *Proceedings - 2015 IEEE International Conference on Healthcare Informatics, ICHI 2015*, pp. 160–169, 2015, Publisher: IEEE ISBN: 9781467395489.

[128] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, May 2019.

[129] N. Akhtar and A. Mian, “Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey,” *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[130] E. Luger and A. Sellen, “”Like Having a Really Bad PA”: The Gulf between User Expectation and Experience of Conversational Agents,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, San Jose California USA: ACM, May 2016, pp. 5286–5297, ISBN: 978-1-4503-3362-7.

[131] R. Kang, L. Dabbish, N. Fruchter, and S. Kiesler, “My Data Just Goes Everywhere: User Mental Models of the Internet and Implications for Privacy and Security,” in *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, Ottawa: USENIX Association, Jul. 2015, pp. 39–52, ISBN: 978-1-931971-24-9.

[132] R. Zhang, N. J. McNeese, G. Freeman, and G. Musick, “”An Ideal Human”: Expectations of AI Teammates in Human-AI Teaming,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW3, pp. 1–25, Jan. 2021.

[133] F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Wortman Vaughan, and H. Wallach, "Manipulating and Measuring Model Interpretability," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, Yokohama Japan: ACM, May 2021, pp. 1–52, ISBN: 978-1-4503-8096-6.

[134] J. J. Cash, "Alert Fatigue," *American Journal of Health-System Pharmacy*, vol. 66, no. 23, pp. 2098–2101, Dec. 2009.

[135] T. Van Gog, L. Kester, and F. Paas, "Effects of concurrent monitoring on cognitive load and performance as a function of task complexity," *Applied Cognitive Psychology*, vol. 25, no. 4, pp. 584–587, Jul. 2011.

[136] J. Sweller, "Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load," *Educational Psychology Review*, vol. 22, no. 2, pp. 123–138, Jun. 2010.

[137] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics Derived Automatically from Language Corpora Contain Human-like Biases," *Science*, vol. 356, no. 6334, pp. 183–186, Apr. 2017.

[138] A. Kuznetsova *et al.*, "The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[139] G. Bansal, B. Nushi, E. Kamar, D. S. Weld, W. S. Lasecki, and E. Horvitz, "Updates in Human-AI Teams: Understanding and Addressing the Performance/Compatibility Tradeoff," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 2429–2437, Jul. 2019.

[140] V. Lai and C. Tan, "On Human Predictions with Explanations and Predictions of Machine Learning Models: A Case Study on Deception Detection," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Atlanta GA USA: ACM, Jan. 2019, pp. 29–38, ISBN: 978-1-4503-6125-5.

[141] T. Wu, D. S. Weld, and J. Heer, "Local Decision Pitfalls in Interactive Machine Learning: An Investigation into Feature Selection in Sentiment Analysis," *ACM Transactions on Computer-Human Interaction*, vol. 26, no. 4, pp. 1–27, Jul. 2019.

[142] M. Yin, J. Wortman Vaughan, and H. Wallach, "Understanding the Effect of Accuracy on Trust in Machine Learning Models," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk: ACM, May 2019, pp. 1–12, ISBN: 978-1-4503-5970-2.

[143] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding Deceptive Opinion Spam by Any Stretch of the Imagination," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*

- *Volume 1*, ser. HLT '11, USA: Association for Computational Linguistics, 2011, pp. 309–319, ISBN: 978-1-932432-87-9.

[144] M. Ott, C. Cardie, and J. T. Hancock, "Negative Deceptive Opinion Spam," 2013, pp. 9–14.

[145] P. L. Teh, P. Rayson, I. Pak, and S. Piao, "Sentiment Analysis Tools Should Take Account of the Number of Exclamation Marks!!!" In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, Brussels Belgium: ACM, Dec. 2015, pp. 1–6, ISBN: 978-1-4503-3491-4.

[146] G. Cheng, J. Han, and X. Lu, "Remote Sensing Image Scene Classification: Benchmark and State of the Art," *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017.

[147] T. Logar, J. Bullock, E. Nemni, L. Bromley, J. A. Quinn, and M. Luengo-Oroz, "PulseSatellite: A tool using human-AI feedback loops for satellite image analysis in humanitarian contexts," *arXiv:2001.10685 [cs, eess]*, Jan. 2020, arXiv: 2001.10685.

[148] P. Welinder *et al.*, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[149] G. Van Horn *et al.*, "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA: IEEE, Jun. 2015, pp. 595–604, ISBN: 978-1-4673-6964-0.

[150] L. Lascau, S. J. J. Gould, A. L. Cox, E. Karmannaya, and D. P. Brumby, "Monotasking or Multitasking: Designing for Crowdworkers' Preferences," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk: ACM, May 2019, pp. 1–14, ISBN: 978-1-4503-5970-2.

[151] E. Glikson and A. W. Woolley, "Human trust in artificial intelligence: Review of empirical research," *Academy of Management Annals*, vol. 14, no. 2, pp. 627–660, 2020.

[152] J. R. Anderson, M. Matessa, and C. Lebiere, "ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention," *Human–Computer Interaction*, vol. 12, no. 4, pp. 439–462, Dec. 1997.

[153] S. Ritter, J. R. Anderson, K. R. Koedinger, and A. Corbett, "Cognitive Tutor: Applied research in mathematics education," *Psychonomic Bulletin & Review*, vol. 14, no. 2, pp. 249–255, Apr. 2007.

[154] S. Barocas and A. D. Selbst, “Big Data’s Disparate Impact,” *SSRN Electronic Journal*, vol. 671, pp. 671–732, 2018.

[155] National Transportation Safety Board, *Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian*, Nov. 2019.

[156] B. Wilson, J. Hoffman, and J. Morgenstern, *Predictive Inequity in Object Detection*, arXiv:1902.11097 [cs, stat], Feb. 2019.

[157] W. H. Deng *et al.*, “Exploring How Machine Learning Practitioners (Try To) Use Fairness Toolkits,” in *2022 ACM Conference on Fairness, Accountability, and Transparency*, Seoul Republic of Korea: ACM, Jun. 2022, pp. 473–484, ISBN: 978-1-4503-9352-2.

[158] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, “Vega-Lite: A Grammar of Interactive Graphics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017.

[159] J. Heer, *Arquero: Query processing and transformation of array-backed data tables*, 2020.

[160] Z. J. Wang, E. Montoya, D. Munechika, H. Yang, B. Hoover, and D. H. Chau, *DiffusionDB: A Large-scale Prompt Gallery Dataset for Text-to-Image Generative Models*, arXiv:2210.14896 [cs], Nov. 2022.

[161] W. McKinney, “Data Structures for Statistical Computing in Python,” Austin, Texas, 2010, pp. 56–61.

[162] A. Krizhevsky, G. Hinton, and others, “Learning multiple layers of features from tiny images,” 2009, Publisher: Citeseer.

[163] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas, and A. Thite, *Jakobovski/Free-Spoken-Digit-Dataset: V1.0.8*, Aug. 2018.

[164] D. Jha *et al.*, “Kvasir-SEG: A Segmented Polyp Dataset,” in *MultiMedia Modeling*, Y. M. Ro *et al.*, Eds., vol. 11962, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 451–462, ISBN: 978-3-030-37733-5 978-3-030-37734-2.

[165] J. Ni, J. Li, and J. McAuley, “Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 188–197.

[166] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Mobile Sensor Data Anonymization," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, ser. IoTDI '19, event-place: Montreal, Quebec, Canada, New York, NY, USA: ACM, 2019, pp. 49–58, ISBN: 978-1-4503-6283-2.

[167] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8693, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10601-4 978-3-319-10602-1.

[168] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[169] J. J. van Griethuysen *et al.*, "Computational Radiomics System to Decode the Radiographic Phenotype," *Cancer Research*, vol. 77, no. 21, e104–e107, Nov. 2017.

[170] T. M. Kolb, J. Lichy, and J. H. Newhouse, "Comparison of the Performance of Screening Mammography, Physical Examination, and Breast US and Evaluation of Factors that Influence Them: An Analysis of 27,825 Patient Evaluations," *Radiology*, vol. 225, no. 1, pp. 165–175, Oct. 2002.

[171] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-Resolution Image Synthesis with Latent Diffusion Models*, arXiv:2112.10752 [cs], Apr. 2022.

[172] D. Bertucci *et al.*, "DendroMap: Visual Exploration of Large-Scale Image Datasets for Machine Learning with Treemaps," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2022, Publisher: IEEE.

[173] P. Moritz *et al.*, *Ray: A Distributed Framework for Emerging AI Applications*, arXiv:1712.05889 [cs, stat], Sep. 2018.

[174] D. Moritz, B. Howe, and J. Heer, "Falcon: Balancing Interactive Latency and Resolution Sensitivity for Scalable Linked Visualizations," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow Scotland Uk: ACM, May 2019, pp. 1–11, ISBN: 978-1-4503-5970-2.

[175] V. S. Chen, S. Wu, Z. Weng, A. Ratner, and C. Ré, "Slice-based Learning: A Programming Model for Residual Learning in Critical Data Slices," no. NeurIPS, 2019.

[176] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, *Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization*, arXiv:1911.08731 [cs, stat], Apr. 2020.

[177] E. Z. Liu *et al.*, “Just Train Twice: Improving Group Robustness without Training Group Information,” *arXiv:2107.09044 [cs, stat]*, Sep. 2021, arXiv: 2107.09044.

[178] OpenAI *et al.*, *GPT-4 Technical Report*, arXiv:2303.08774 [cs], Dec. 2023.

[179] Z. J. Wang, A. Chakravarthy, D. Munechika, and D. H. Chau, “Wordflow: Social Prompt Engineering for Large Language Models,” *arXiv 2401.14447*, 2024.

[180] A. Cabrera, M. Tulio Ribeiro, B. Lee, R. Deline, A. Perer, and S. M. Drucker, “What Did My AI Learn? How Data Scientists Make Sense of Model Behavior,” *ACM Trans. Comput.-Hum. Interact.*, vol. 30, no. 1, Mar. 2023, Place: New York, NY, USA Publisher: Association for Computing Machinery.

[181] B. Yu, Y. Yuan, L. Terveen, Z. S. Wu, J. Forlizzi, and H. Zhu, “Keeping designers in the loop: Communicating inherent algorithmic trade-offs across multiple objectives,” *DIS 2020 - Proceedings of the 2020 ACM Designing Interactive Systems Conference*, pp. 1245–1257, 2020, ISBN: 9781450369749.

[182] W. H. Deng, N. Yildirim, M. Chang, M. Eslami, K. Holstein, and M. Madaio, “Investigating Practices and Opportunities for Cross-functional Collaboration around AI Fairness in Industry Practice,” in *2023 ACM Conference on Fairness, Accountability, and Transparency*, Chicago IL USA: ACM, Jun. 2023, pp. 705–716, ISBN: 9798400701924.

[183] S. N. Akter *et al.*, “An in-depth look at gemini’s language abilities,” *arXiv preprint arXiv:2312.11444*, 2023.

[184] Z. Zhang *et al.*, “Understanding Business Analysts’ Needs for Data Report Authoring,” 5 pages, 2022, Artwork Size: 5 pages Publisher: The Eurographics Association.

[185] A. Satyanarayan *et al.*, “Critical reflections on visualization authoring systems,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 461–471, 2020.

[186] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, “Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 649–658, 2016.

[187] K. Wongsuphasawat *et al.*, “Voyager 2: Augmenting visual analysis with partial view specifications,” *Conference on Human Factors in Computing Systems - Proceedings*, vol. 2017-May, pp. 2648–2659, 2017, ISBN: 9781450346559.

[188] D. Ren, B. Lee, and M. Brehmer, "Charticulator: Interactive Construction of Bespoke Chart Layouts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 789–799, 2019.

[189] C. Stolte and P. Hanrahan, "Polaris: A system for query, analysis and visualization of multi-dimensional relational databases," *Proceedings of the IEEE Symposium on Information Visualization*, pp. 5–14, 2000.

[190] G. Team *et al.*, *Gemini: A Family of Highly Capable Multimodal Models*, arXiv:2312.11805 [cs], Dec. 2023.

[191] L. Gao *et al.*, *A framework for few-shot language model evaluation*, Version Number: v0.4.0, Dec. 2023.

[192] *OpenLLM Leaderboard*.

[193] *RAGAS*.

[194] K. Cobbe *et al.*, *Training Verifiers to Solve Math Word Problems*, arXiv:2110.14168 [cs], Nov. 2021.

[195] A. Q. Jiang *et al.*, *Mixtral of Experts*, arXiv:2401.04088 [cs], Jan. 2024.