Using Visual Analytics to Support Artificial Intelligence Development

Nan-Chen Chen

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Cecilia Aragon, Chair

Been Kim

Gary Hsieh

Amy Ko

Program Authorized to Offer Degree:

Human Centered Design & Engineering

University of Washington

Abstract

Using Visual Analytics to Support Artificial Intelligence Development

Nan-Chen Chen

Chair of the Supervisory Committee:
Cecilia Aragon
Human Centered Design & Engineering

In the artificial intelligence (AI) era, complex AI is becoming more common in daily life. Various reports indicate that AI plays an increasingly influential role in human society (e.g., [Russell et al. 2015, Stone et al. 2016]). However, the development of a sophisticated AI system is a challenging process. AI researchers must experiment with different AI system architectures and evaluate their performance on the target tasks. As components in the system interact with each other, it can be difficult to make sense of evaluations and find insight to improve the system. Traditional metrics for system performance (e.g., task accuracy) often fail to provide useful and actionable insights for AI developers. The burden for AI researchers is to analyze evaluations to understand how changes impact system behavior, and then find ways to enhance the system.

Visualization is a powerful tool to present information in a way that is easier for humans to absorb than text [Little 2015]. Given that AI development involves a vast amount of information, visualization can facilitate insight discovery for AI developers. Among all possible visualization approaches, *visual analytics* is that which specifically focuses on using visualization to support analytical tasks and is a powerful method for insight discovery. However, designing visual analytics tools is not a trivial task. In particular, since AI development is highly specialized and requires a significant amount of experience, designing visual analytics tools to satisfy AI developers' needs is challenging. Human-centered design is a powerful technique to address such needs. Therefore, in this thesis, we focus on using a human-centered design approach to study, design, and engineer visual analytics tools to support AI system development.

The thesis includes the design of two visualization tools: *QSAnglyzer*, a visual analytics tool for evaluation analysis; and *AnchorViz*, an interactive visualization for discovering errors in interactive machine learning classifiers. In addition, based on an interview study, we propose a framework to describe the current practice of AI development workflows, highlighting issues and suggesting design implications for designers and researchers. By using a human-centered design approach, this thesis aims to contribute to the fields of human-computer interaction (HCI), visualization (VIS), and visual analytics (VA) about AI developers and AI development processes, as well as how to design visual analytics for this domain.Our ultimate goal is to pave a road for creating better tools for AI developers, lowering the barriers for AI development, and making AI more accessible to a wider range of people to build and use.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I have been waiting for the moment when I would be able to write this section ever since I started my life as a PhD student. Now is finally the time, but I find it almost impossible to express my thankfulness in such a limited space. If I were to thank everybody who has helped me, my acknowledgment section would be as long as my thesis. Therefore, please understand that what I write in this section is just a small portion of my gratitude to all the people who have helped me.

First, I would like to thank my advisor, Cecilia Aragon, who has led me into the world of visual analytics (VA) and ethnographic studies. Without Cecilia, I would not have stepped into the human side of technology as deeply as I have. In fact, without Cecilia, I would not have come to Seattle and University of Washington (UW) for my PhD study, which is the best decision of my life.

I still remember the day I met Cecilia in person in 2013. It was a beautiful March day in Seattle, but a shadow was on my mind since I was unsure whether I wanted to pursue a PhD degree in an interdisciplinary department, not a pure computer science (CS) department. I used to dream of being a professor in CS in Taiwan, and thus I had strong concerns about whether an interdisciplinary PhD degree might block the path to my dream.

The truth is that at the time I applied to Human Centered Design & Engineering (HCDE), I did not understand that HCDE is a department. I had gone to the Design, Use, and Build (DUB) group website (because I thought DUB was a research lab) and found a page named “how to join us.” HCDE was listed at the top of the page at the time. Given that I had seen many programs specializing in human-computer interaction (HCI) listed under CS, I thought HCDE would be the same. It was not until I was admitted that I realized that HCDE is actually a department, not just a program!

As a prospective student visiting UW and Seattle for the first time, I immediately fell in love with the city and the gorgeous campus. But I was not sure if I should step out of my comfort zone in CS, in which I had been immersed for more than 20 years. Before meeting Cecilia, I stood in front of the DUB lawn, looking at the Allen Library and asking myself, “Should I come here for my PhD?”

It was not until I met Cecilia that I made up my mind to come. At the beginning of the meeting, we greeted each other and chatted for a while. Cecilia then asked me to write code on her whiteboard to reverse a string. To many people, that may sound scary, but to me, it was a fun activity, especially since I knew she was the creator of the Treap, a data structure that I learned when I was still a high school student participating in programming contests. In my meeting with her, Cecilia made me understand how many opportunities an interdisciplinary degree could bring; given that I had demonstrated that I am a good programmer, I did not need to worry about

not being able to become a professor in CS. As a result, I decided to go on an adventure in the unknown world of HCDE.

I could write a dozen pages of stories about working with Cecilia, but to cut a long story short, Cecilia brought me into the world of visual analytics and ethnography. I was unsure about their usefulness and importance at the beginning and constantly worried about my research not being sufficiently "technical" compared to work on the latest fashionable technology. In addition, as a non-native English speaker and international student, I did not have the confidence to do ethnography well. But visual analytics and ethnography, as well as the human-centered design processes that I learned from carrying out research with Cecilia, turned out to be the backbone of this thesis. When a study participant told me that he greatly liked the visual analytics tool I had built, even though he had questioned its usefulness when I started the project, I could not adequately describe my appreciation for how Cecilia had cultivated my development of this unique skill set. I will be grateful to Cecilia as I continue to leverage this skill set throughout my career.

Another key person who has made this thesis happen is Been Kim. Been is my committee member as well as my internship mentor at the Allen Institute for Artificial Intelligence (AI2). I met Been when she gave her first talk at the DUB seminar at UW in October 2015. I was actually late for the talk and almost decided not to go. However, since Cecilia had recommended the talk (thank you, Cecilia!), I felt that I should at least check it out. Through Been's talk, I discovered that her work could be used in a research project I was working on. Thus, I had a conversation

with her after her talk and showed her the visual analytics tool I was working on. Been was very interested in the tool, as well as the research project I was working on. As a result, she asked me whether I was interested in applying for an internship at AI2 to work with her. I applied, and that internship led to the creation of QSAnglyzer and launched my thesis direction—building visual analytics for artificial intelligence (AI) developers.

Been is unbelievably amazing. Although her background is in AI and machine learning, she cares about people and appreciates HCI work. The internship with her was one of the best time periods in my entire PhD student life. We carried out user studies together and had countless passionate discussions about how to make things right. I was able to experience many positive intellectual challenges that helped me grow very fast. Been is also very supportive. Even though she was very busy, every day she would come to me and say, “How are things going?” and give me much constructive feedback. In addition, Been is an excellent listener with a sense of humor who is very good at rephrasing things in a positive tone. Thinking back, I realize that she turned the many disagreements we had into meaningful discussions rather than emotional conflicts. Not everyone can do this well, myself included. I really appreciate how much Been has done for me. Her support has continued even after the internship and even after she left AI2. It has been my great pleasure and luck to have had her on my committee and now as a colleague at Google. In fact, I am working in a team that builds products for explainable AI (XAI), which is her research topic. Of course, I got to know this team because of Been.

I also want to thank Lavanya Ramakrishnan for supporting my PhD student life. Lavanya is a scientist at the Lawrence Berkeley National Lab (LBL). She is also the principal investigator (PI) of the ethnographic studies project through which Cecilia funded me from 2014 to 2017. That project is *the* project that forced me to learn how to do ethnographic studies. It is also the project that made me understand the importance of focusing on aspects of workflow. My first trip to Berkeley started by missing a flight. The next day, when I went to LBL for the first time, I was asked to carry out an interview with the scientists we wanted to study on their workflows using supercomputers. I did not know I was supposed to conduct the interviews, and I had only had three hours of sleep the day before due to missing my flight, so I was very nervous. But luckily, Lavanya and Cecilia, together with Sarah Poon (a user experience (UX) researcher at LBL) and Brittany Fiore-Silfvast (Cecilia's postdoc at the time), helped me during the interviews.

My three years working on the LBL project completely changed the way I view and interact with the world. Through this project, I gradually learned how to ask good questions that can bring to the surface the issues that people encounter. I became a person that can talk to strangers, and I started enjoying such conversations, even though I used to be very shy. The two papers we published for the project were big challenges for me since I had never written papers that followed social science traditions. Lavanya and Sarah helped along the way. Lavanya supported me not only intellectually but also emotionally. On every trip to LBL, I felt welcomed. In those three years, I experienced many struggles, but with support from Lavanya, I became better and better at doing field work. In addition, Lavanya spent a lot of time helping me improve my

writing. When I wrote Chapter 5 of my thesis, I constantly recalled things I learned from the two papers and realized how much better I had become at writing. Even after the project ended, Lavanya still showed her care for me and my work. The LBL project is not part of this thesis, but the things I learned from the project, and the support I received from Lavanya, were indispensable in making this thesis happen.

I also very much want to thank Jina Suh. Jina was my internship mentor at Microsoft Research (MSR) in 2017, and we created AnchorViz together. Before the internship, we had been collaborating in Cecilia's TextVisDRG. Working with Jina is always an enjoyable experience. She is extremely diligent, easy to work with, and creative. She thinks critically and asks crucial questions. I can always sharpen my ideas through talking to her. She is also an experienced and skillful developer. I cannot describe how much fun it is to write code together with Jina. The idea of AnchorViz emerged very late during my summer internship, so we only had a few weeks to finish the project, including its implementation and evaluation. It was really intense, but I had so much fun. Without her, Chapter 4 of my dissertation would not exist.

Another person in MSR I want to thank is Gonzalo Ramos. Gonzo was my internship mentor in my second internship at MSR, but we had worked together since my first internship. I respect Gonzo a lot for his passion for research as well as how much he is willing to help others. Gonzo is always like a superhero who is willing to protect and save others selflessly. Even after my internship was over, Gonzo constantly showed his willingness to help me and support me. I had a

lot of struggles during the last year of my PhD study, and Gonzo's support played an important role in helping me to overcome this difficult time.

I want to thank my committee members, Amy Ko, Gary Hsieh, and Jason Yip. I am very lucky to have had a great committee that has been very supportive and provided me with constructive feedback on my proposal and thesis. I especially want to thank three other faculty members: Charlotte Lee, Daniela Rosner, and Jennifer Turns. Charlotte was another person who confirmed my decision to come to HCDE, and she taught me the ethnographic methods that enabled me to carry out the LBL project. Daniela taught me design thinking and visual communication, which are valuable skills from which I continue to benefit. Her words to me, "The way we think about the world will change the way we shape the world," are my favorite HCDE motto. Jennifer taught a class I attended on the empirical traditions of HCDE, which gave me an overview of different epistemologies. I treasure a number of conversations we had as part of this class.

I also want to thank my labmates: Michael Brooks, Daniel Perry, Ray Hong, Margaret Drouhard, and Rafal Kocielnik. The journey we have taken together in the world of visual analytics has been unforgettable. I am also grateful to my two Taiwanese PhD friends in HCDE, Christina Chung and Ying-Yu Chen, who arrived a year before me and accompanied me on most of my PhD journey. They taught me many things, including important life lessons. In addition, I want to thank three people who used to be in my PhD cohort: Yi-Chen Sung, Pam Munro, and Rachel Ulgado. Even though we did not go through the whole journey together, they are the ones

who shook my original biased beliefs about humans and made me constantly rethink the relationship between humans and technology. Furthermore, I want to thank my editor, Aaron Heidel, for helping edit my writing. Without Aaron, this thesis would not have been possible.

In addition to the people at UW, my journey as a PhD student would not have reached the end without support from AI2, MSR, Hsing Tian Kong, and Google. AI2 made two-thirds of my thesis possible. I especially want to thank the Aristo team and my co-author Kyle Lo. I am extremely lucky to have had the opportunity to work in an AI research institute as an HCI researcher. MSR awarded me the Microsoft Research Graduate Women's Scholarship in 2014, which was a great affirmation when I had just got started. The machine teaching group at MSR made me believe I could do good work and gave me the chance to work in a world-class research institute. In addition to Jina and Gonzo, Patrice Simard, Johan Verwey, Soroush Ghorashi, and Steven Drucker are the key people who made my internships invaluable. Hsing Tien Kong gave me a scholarship for eight years, enabling me to live my PhD life without worries. Google awarded me the Anita Borg Scholarship, and I am now extremely lucky to work in the XAI team at Google Cloud, applying all of my PhD skillset and experience in the real world.

I would not have been able to start my journey without three people: my undergraduate advisor Hao Chu, my scholarship funder L.T. Wang, and Prof. Randy Pausch. Hao led me into the world of HCI research. Without my research experience as an undergraduate in UbiComp, I would not have had the chance to come to the U.S. for my PhD. L.T. not only gave me scholarship to support my life expenses but also helped me edit my PhD applications. L.T. also

provided support and help when I first came to the U.S. and when I was at Berkeley. Without Prof Pausch's book *The Last Lecture*, I would not have decided to pursue a PhD in HCI when I was an undergraduate.

Numerous people have given me support on my journey. My family—my father Ming-Tong, my mother Tsuey-Ling, my sister Yi-Hsuan, my brother Wei-Yu, and my nanny Chin-Lien—have given me love and support and enabled me to come to the U.S. for study. My Taiwanese friends in the U.S., Pei-Fen Tu, Chia-Wei Lien, Ping-Chia Tsai, Jennie Yi Chen, Ying-Chin Chen, Ching-Hsia Cheng, and Hsin-Yu Lu, are the best listeners and my family in the US. Together, they have helped me face my inner self and supported me to overcome the difficulties in the past six years. My friends in Taiwan, Verena Chien, Jenny Chen, and Jush Lu, are the most influential people in terms of the way I view the world. My first internship mentor ever in life, Erin Fitzhenry, gave me the courage to face challenges in the world and accompanied me in a critical time when I had a lot of questions about research and the world.

Last but not least, I want to thank my lovely cat, which has accompanied me for the entire thesis writing process. I once lost the meaning of life and was unsure what I was doing. But my cat made me feel like a human again. She made me understand many things I did not understand in my earlier years and made me reflect on the privileged and judgmental way in which I think. The long journey of pursuing my PhD is finally coming to an end, but my real-life journey is about to start. With my cat, I know I will have the courage to use all the things I have learned from my PhD study to make the world a better place.

DEDICATION

This thesis is dedicated

to those who insist on humanity

in an era of artificial intelligence.

Chapter 1. Introduction

In the artificial intelligence (AI) era, complex AI systems such as conversational agents, self-driving cars, and recommendation systems are gradually becoming part of daily life. Various reports indicate that AI plays an increasingly influential role in human society [Russell et al. 2015, Stone et al. 2016]. Major technology companies [Bell 2016, Lewis-Kraus 2016] as well as government agencies [DARPA 2016] are putting increasing emphasis on AI applications.

The development of a sophisticated AI system is a challenging process. AI researchers experiment with different AI system architectures and evaluate their performance on the target tasks. As components in the system interact with each other, it can be difficult to make sense of evaluations and find insight to improve the system. Take question answering (QA) systems as an example: A QA system is a type of AI system that can automatically produce answers to a given question in human language. Such a system may involve information retrieval (IR), natural language processing (NLP), knowledge representation, reasoning, and machine learning (ML) components. All components are complex in their own ways, and can interact with each other in complicated ways such that the impact of changes in one component can propagate to overall changes in behavior. AI researchers may modify one component based on insight gained from one set of evaluations, but sometimes discover that the overall behavior of the system changes in unexpected ways. For instance, an NLP component can take the output of an IR component as input, and pass this onto an ML component. Small changes in the IR component may thus propagate to the ML component, and lead to unanticipated changes in overall behavior.

In more recent years, the rise of deep learning has changed the field of AI significantly. For example, *AlphaGo*, a deep-learning-based AI system for the board game Go, successfully beat world champion Go players in 2016 and 2017, rewriting the history of AI which for 30 years could not beat even amateurs [Fogg 2017]. The use of deep learning has also significantly increased the accuracy on tasks such as computer vision, machine translation, and speech recognition. Although a variety of AI systems have become available, it is still challenging to develop an AI system, especially given that deep-learning-based AI systems are usually difficult to understand.

Traditional metrics for system performance (e.g., task accuracy) often fail to provide useful and actionable insights for AI developers. For instance, two versions of a QA system can both achieve 50% accuracy by answering completely different sets of questions correctly. One version may answer all the questions starting with ‘what’ correctly, whereas the other version may get all the ‘why’ questions right. More complex patterns of the strengths and weaknesses of each version are challenging to discover and track. The burden for AI researchers is to analyze evaluations to understand how changes impact system behavior, and then find ways to enhance the system.

Visualization is a powerful tool to present information in a way that is easier for human to absorb than text [Little 2015]. Visualization can also help reveal patterns and trends in data to make discovery easier [Chibana 2017]. As far back as the mid-1850s, John Snow used a map to visualize the spread of the cholera epidemic in London and discovered that the disease was spread through the sewer system [Sandberg 2013]. Since then, the use of visualization as an analytical tool has shown its success in fields such as physics, climate science, finance, and

biology. Such work focusing on designing visualization tools to support analytical tasks has gradually become a field in its own right, termed *visual analytics* [Cook and Thomas 2005].

Given that AI development involves a vast amount of information, visualization can shed light on model behavior, off-loading burdens from AI developers and facilitating insight discovery. Among all possible visualization approaches, visual analytics is that which specifically focuses on using visualization to support analytical tasks and is a powerful method for insight discovery. However, designing visual analytics tools is not a trivial process. In particular, as AI development is highly specialized and requires significant experience, designing visual analytics tools that satisfy AI developers' needs is challenging. Therefore, in this thesis, we focus on using a human-centered design approach to study, design, and engineer visual analytics tools to support AI system development.

According to the International Standardization Organization (ISO), human-centered design is *"an approach to interactive systems development that aims to make systems usable and useful by focusing on the users, their needs and requirements, and by applying human factors/ergonomics, usability knowledge, and techniques. This approach enhances effectiveness and efficiency, improves human well-being, user satisfaction, accessibility and sustainability; and counteracts possible adverse effects of use on human health, safety and performance [ISO 2009]."* By using a human-centered design approach, this thesis aims to contribute to the knowledge in the field of human-computer interaction (HCI), visualization (VIS), and visual analytics (VA) communities about AI developers and AI development processes, as well as how to design visual analytics for this domain. My ultimate goal is to pave a road for creating better

tools for AI developers, lowering the barriers for AI development, and making AI more accessible to a wider range of people to build and use.

1.1 Research Questions

In this thesis, we choose to focus on answering the following research questions:

RQ1. What are the challenges AI developers encounter in analyzing evaluation results?

RQ2. How can we design visualization tools to support evaluation analysis?

RQ3. How can we design visualization tools to facilitate interactive machine learning?

RQ4. What processes and issues do AI developers have in building AI systems?

We answer the above research questions through three studies. In Chapter 3, we describe a detailed study on QA system evaluation analysis for answering RQ1 and propose *QSAnglyzer* for answering RQ2. In Chapter 4, we switch the focus to an interactive development paradigm and propose a visualization tool named *AnchorViz* for error discovery, which aims to answer RQ3. Finally, in Chapter 4, we conduct an interview study for RQ34 to extract key elements, tasks, and challenges in more diverse AI system development scenarios, and propose a framework to describe these key elements, tasks, and challenges in AI development workflows.

These research questions were formed after initial interactions with AI developers, which covered two specific aspects of AI development: evaluation analysis and error discovery in interactive machine learning. These two specific aspects are key to AI development. Evaluation analysis is a task that AI developers perform to understand their systems' current behaviors and performance so that they can come up with potential ways to improve the systems or define the limitations of the systems. Chapter 3 will provide a more detailed explanation about the goals of

evaluation analysis, and will illustrate how to extract tasks and requirements that are used to design QSAnglyzer.

Similarly, error discovery is a critical task in interactive machine learning (iML), a new paradigm of machine learning that emphasizes putting humans in the loop when building ML models. Unlike traditional ML, which uses a pipeline approach, iML allows AI developers to iteratively label data, add features, train models, and evaluate to see current results. The benefit of such an approach is to ensure that what models learn is aligned with what AI developers want them to learn. However, since datasets are iteratively labeled, it is more difficult to discover errors in the unlabeled set in iML. Therefore, in Chapter 4, we propose leveraging interactive visualization to mitigate this difficulty. In AnchorViz, users can create “anchors” to represent concepts they have discovered in the datasets, and use these anchors to slice and dice the datasets to find inconsistencies between labeled items and predicted labels of unlabeled items.

The above two research directions are specific but important aspects of AI development. However, more general research on the development process is needed. Therefore, the last research question focuses on gaining an overview of AI development, and on extracting challenges in AI development workflows. We aim to highlight potential opportunities for future researchers, as well as illustrate the contexts of AI development to scaffold future designers who seek to design for AI developers. Thus, in Chapter 5, we leverage an interview study to come up with a framework to describe AI development processes, and report challenges and opportunities in current workflows.

According to Wobbrock & Kientz (2016) [Wobbrock and Kientz 2016], research contributions to the HCI field can include empirical findings, artifact contributions, and

theoretical contributions. In this thesis, the empirical findings include all the user studies we conducted for the tool design and the interview study. The artifact contributions include both the QSAnglyzer design and the AnchorViz design. The theoretical contributions include the framework we extract from the interview study. As AI becomes more and more influential in human life, we hope the contributions from this thesis will foster more research in this direction and ensure that we tackle the issue from a human-centered design and engineering approach.

1.2 Thesis Statement

Visual analytics (VA) is a powerful approach to support data exploration and analysis in AI development workflows. By leveraging visual representations and interaction techniques, analytical tasks in AI development such as comparing experiment results and discovering errors can be accelerated. In addition, results from these analytical tasks are often used to inform other tasks in AI development workflows such as modifying AI system design. Therefore, a framework of AI development workflows can help designers to identify opportunities for VA tools and better plug VA tools into existing workflows.

Chapter 2. Background

Artificial intelligence (AI) is one of the earliest research fields in computer science. Since the field was formally established in 1956, researchers have been trying to develop machines that can demonstrate cognitive capabilities and perform tasks like human beings. The research field has experienced a number of ups and downs as people have been optimistic about the field, only later to find the progress blocked by the limitations of the time. It was not until recent decades that the AI field has had another rise due to the progress in machine learning (ML) research and the advance of computation hardware (e.g., graphics processing units, *GPU*).

Machine learning (ML) is a powerful approach which builds statistical models based on data to perform human-specified tasks. Unlike traditional AI systems, which are mostly rule-based or knowledge-base (KB)-based, ML trains models to best fit the data, and uses the trained data to make predictions. In the 2000s, ML methods such as support vector machines (SVMs), logistic regression, decision trees, and k-means found wide use in different fields. Later on, artificial neural networks (ANNs or simply *NNs*), a method proposed as early as 1943, became feasible after advances in computing hardware in the late 2000s. In 2009, Nvidia showed how NNs can be trained on GPUs with speeds up to 100 times faster. This made it possible to train *deep neural networks* (DNNs, a type of NNs that emphasizes using many layers of neurons to learn) within a few days. The success of DNNs in many applications like image recognition has made NNs popular again and they have come back with a new name: *deep learning.*

Deep learning fundamentally changed how AI systems are designed (hereafter, we refer to ML methods like SVMs as *traditional ML methods*, as distinct from deep learning methods). To be more specific, in traditional ML-based AI systems, components interact with each other in a

pipelined manner, but deep learning enables *end-to-end* learning). For example, in traditional QA systems, a system usually includes components for language processing, text entailment (an NLP task that infers a directional relation between segments of text), and reasoning. Each component is designed separately and run step-by-step. Some components are not necessarily ML-based, and components may not be tuned together. Therefore, when AI developers change component A based on an error on one data point, component B, another component that takes the result of the component A, may not be changed accordingly. As a result, since the input to component B changes, it can lead to more errors in component B's results. In contrast, in deep learning, AI developers try to design the NNs to be an *end-to-end* system. Namely, instead of having separated components, each of the components is represented by a few layers of neurons, and layers are connected. When the network is learning, the output from the previous layer is fed into the next layer as input, and the parameters are updated accordingly. Thus, there are no longer inconsistencies between components [Roza 2019]. In addition, in traditional ML, features are designed and engineered directly by humans, whereas in deep learning, it is the network architecture that is designed, and features are learned directly by NN models.

As the result of advancements in ML and NN, AI in modern language is usually used to refer to ML-based AI systems. In this thesis, we use "AI" as an umbrella term that includes both early rule-/knowledge-base (KB)-based AI systems as well as ML-based AI systems. The "model" term usually refers to ML models. In Chapter 5, the boundary of the two terms will be further blurred as AI developers now focus on developing ML-based AI systems.

The usefulness of AI has been proven in a variety of applications; however, developing an AI system is still a challenging task and requires the close involvement of AI experts. As a result,

the use of AI is usually limited to very specific tasks, which are not always available to the general public [Amershi et al. 2014]. Furthermore, even with experienced AI practitioners, engineering a “good-enough” AI system for real-world purposes is not a trivial task, particularly as it is still difficult for AI systems to take domain knowledge into account [Sacha et al. 2016, Ware et al. 2001].

Therefore, the challenge of developing AI systems is not merely on the algorithm side. Instead, it is important to consider the development processes from a human-computer interaction (HCI) perspective. For instance, what is a more intuitive way for AI developers to tune their models? How are we to avoid human errors during data processing or labeling steps? How are we to make comparison between evaluation results easier? These questions show why human-related aspects are critical to improve AI development processes.

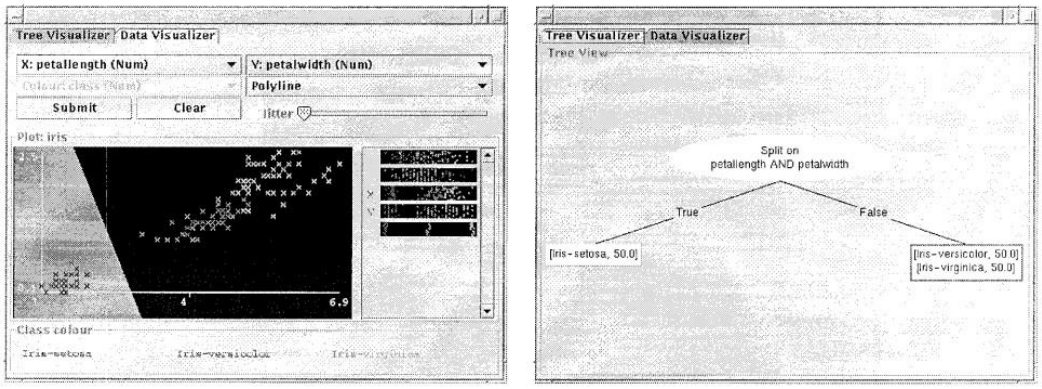

Figure 2.1: Ware et al.’s interface for building decision trees [Ware et al. 2001]

In response to these challenges, various ways have been suggested to rethink the relations between humans and machines in AI tasks. A number of variations on traditional ML methods have been proposed. For example, Ware et al. describe an interface for users to build decision

trees interactively [Ware et al. 2001]. They point out the importance of engaging users when building classifiers and utilize the interface to provide essential information for users to make judgments. Fails and Olsen propose a framework for interactive ML and describe *Crayons*, an interactive tool which embodies their framework in letting end-users construct classifiers for image processing. More specifically, users can interactively examine the results and mark areas on the images to indicate errors [Fails and Olsen Jr 2003]. In their user evaluation, participants with little background in image processing were able to build classifiers within only a few iterations. Amershi et al. further highlight the importance of considering humans in ML development workflows. They point out that people are not oracles (i.e., someone who passively provides ground-truth labels) [Amershi et al. 2014, Baum and Lang 1992]. Labels provided by people can be impacted by the humans' existing knowledge and experiences [Amershi et al. 2014]. In addition, transparency can foster better understanding of the systems and also help people provide better labels.

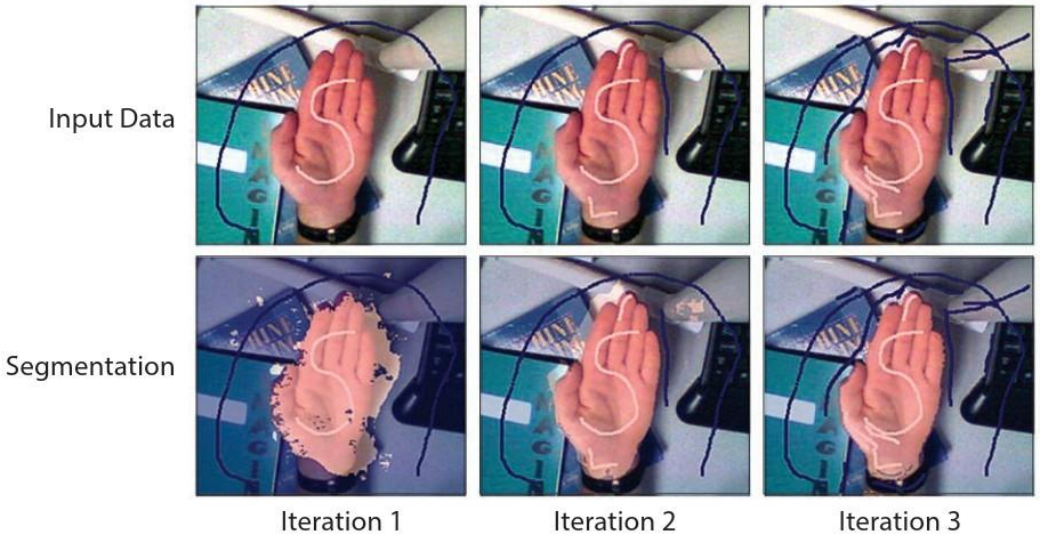

Figure 2.2: Crayons

Amershi et al. also review a few novel interfaces that support user interaction with ML which aim to incorporate human knowledge more effectively and efficiently and increase the interpretability of ML outputs. Aligned with this vision, Kim proposes a framework to consider human-machine collaboration by emphasizing communication [Kim 2015]: To better support collaboration between human and machines, one can either enhance communication from machine to humans by providing more intuitive explanations of the outputs and/or the machines' internal states, or improve communication from humans to machines by incorporating feedback from humans.

The focus of this thesis work is on building tools to support AI development workflows. Previous work on this topic can be divided into two directions: enabling interactive ML and supporting ML model development. The two directions overlap with each other. The following subsections review literature in each direction and discuss the connection between the two directions.

2.1 Enabling Interactive Machine Learning

The thread of work for supporting AI development starts from building tools to enable interactive machine learning (*interactive ML* or *iML*). Specifically, the work focuses more on traditional ML rather than deep learning ML since it is still computationally challenging for deep learning models to be interactive.

Unlike classical pipeline-style ML which focuses on feature selection and the training process, interactive ML puts more emphasis on the feedback loop from users [Ware et al. 2001; Kim 2015]. One early direction on this topic was interactive construction of decision trees, of which Ankerst et al. and Ware et al.'s work was the most well-known [Ankerst et al. 1999, Ware

et al. 2001]. These two papers provided an interface for users to dynamically create the splitting points of a decision tree; the former supports only univariate splitting, whereas the latter allows bivariate splitting. Both works leverage visualization to show variable distributions and classification performance to facilitate users investigating and deciding whether to add more splitting points. Although this approach is promising, it largely depends on the fact that decision trees are easier for humans to understand than other ML algorithms. Indeed, extending the framework to classification algorithms or other ML problems is not a trivial task. Moreover, to be effective enough for given tasks, some decision trees require more than a thousand splitting points; humans are unlikely to able to create this from scratch.

This brings up an interesting question for the interactive ML research community: How much human input is required to build models that are "good enough"? On the other hand, how much human input should the model take into account with regard to the information from data? This is related to Kim's consideration in building an interactive Bayesian model [Kim 2015]. To balance user feedback with information from data, Kim proposes a Bayesian model to take into account user confidence on the feedback.

2.1.1 *More diverse settings for interactive classification*

In more recent years, more diverse directions of interactive ML have emerged. One key direction remains focused on classification, but with greater emphasis on the various settings. For example, Talbot et al. propose *EnsembleMatrix* [Talbot et al. 2009], a system supporting the construction of ensemble classifiers (multiple sub-classifiers with different settings, where combinations are used to determine the final outputs).

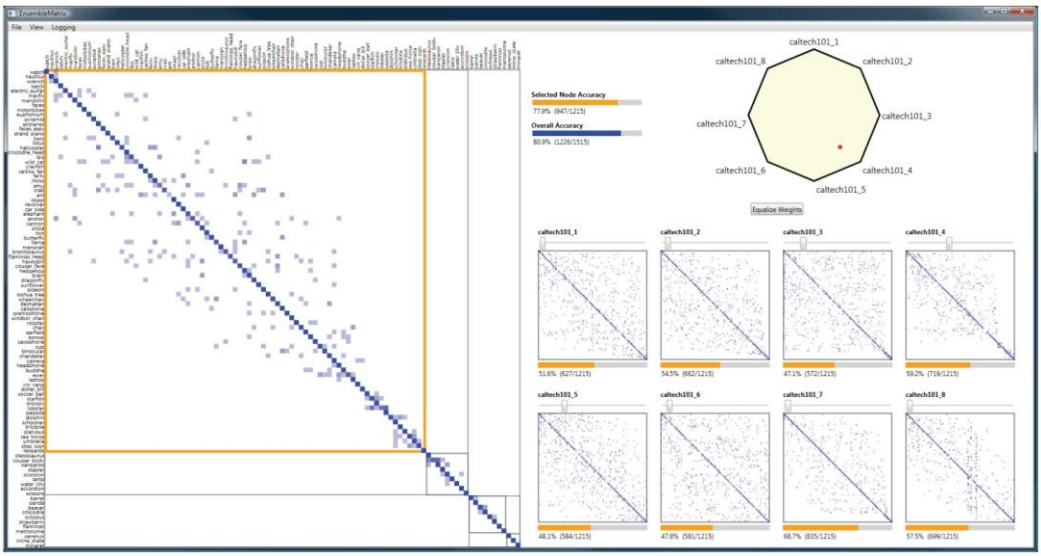

Figure 2.3: EnsembleMatrix

In EnsembleMatrix, users interact with a confusion matrix to visually examine the relations between classes. Heuristics are used to sort classes so that classes with higher mis-classification rates are shown first. Users specify a line on the matrix to create sub-classifiers. Furthermore, they show the performance of each sub-classifier on the right, and enable fine-tuning of combination weights. In their user study, which evaluates only the weight-tuning part, participants create ensemble classifiers that are close to the highest accuracy on record. Despite encouraging results from this study, it is difficult to say how suitable this method is for other classification tasks, as the eight sub-classifiers prepared by the authors of the study were already at about 50% accuracy, which is relatively high for a multi-class classification problem. For some classification tasks, it may be more difficult to create sub-classifiers that work well enough. Moreover, since the confusion matrix visualization should be re-ordered after the weights are updated, keeping track of the creation of sub-classifiers becomes challenging. Furthermore, since

features can strongly influence classifier behaviors, integrating feature-related interaction into the workflow in EnsembleMatrix remains an open problem.

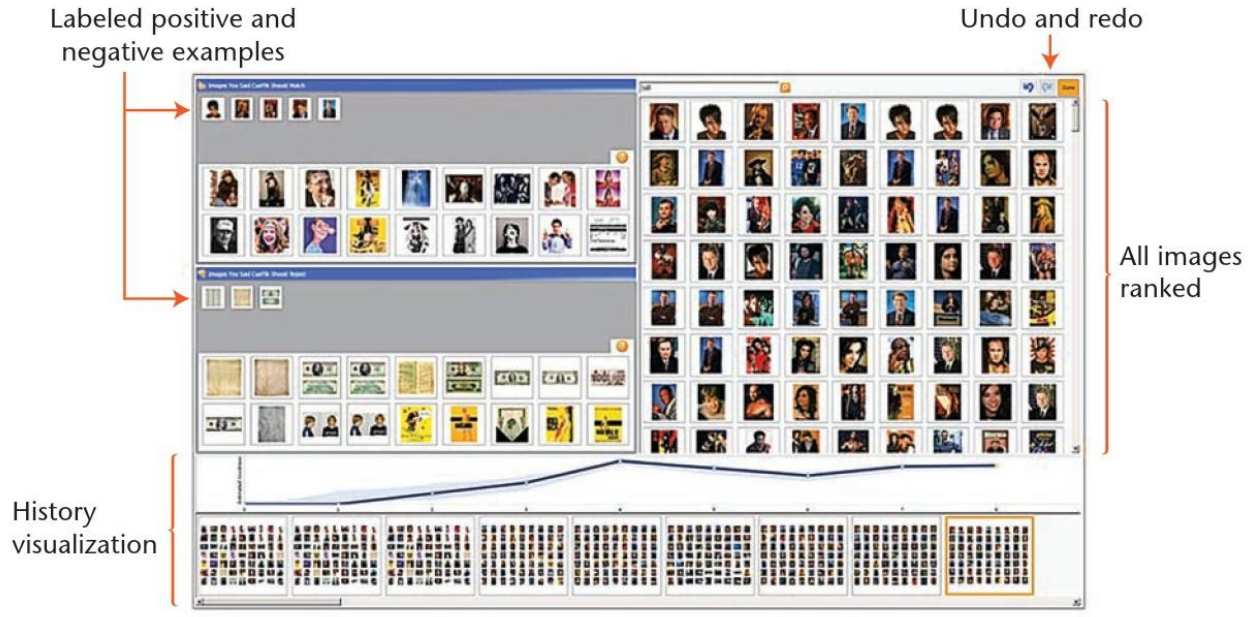

Figure 2.4: CueFlik

Another example for interactive classification is *CueFlik*, Amershi et al.'s system for image classification [Amershi et al. 2011]. In CueFlik, users provide additional positive and negative data points for training a target concept. The system selects representative examples for positive and negative classes; based on these examples, users decide which images they want to supply to the system. The progress of the accuracy over time is presented in a visualization on the interface. While their study shows that the accuracy of the models improves over iterations, they also find little benefit from selecting examples using an active learning approach (i.e., selecting data

points that have the most information gain). Similarly, it may be difficult for users to know which examples are “good examples” that will help improve accuracy.

2.1.2 *Human-intuitive ways of input*

Another direction for enabling interactive ML is altering ML models based on human preferences. More precisely, work in this direction involves providing human-intuitive ways for user input, and then making ML models that take these inputs into account.

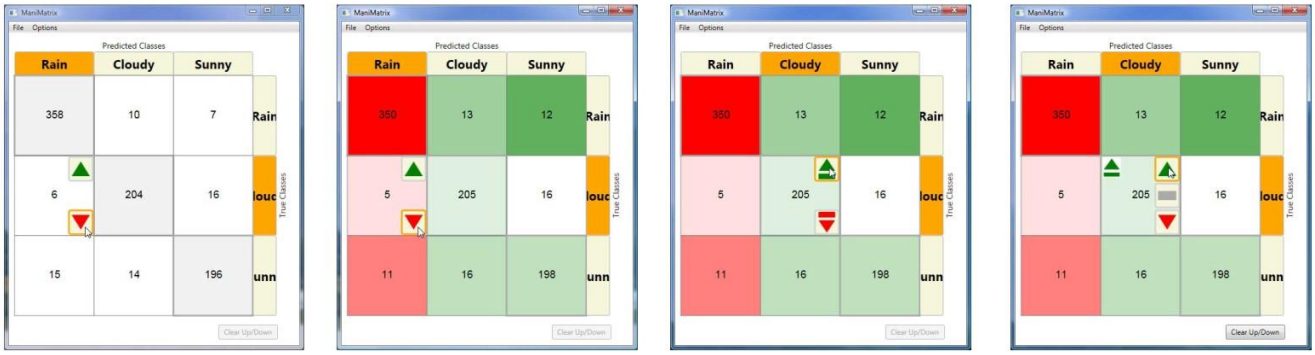

Figure 2.5: ManiMatrix

For instance, Kapoor et al. describe *ManiMatrix* [Kapoor et al. 2010], a system in which users specify target confusion matrices in multi-class classification setups. More specifically, users control how many instances are misclassified from Class A to Class B by increasing or decreasing tolerance numbers based on current numbers. Then the system attempts to find decision boundaries that fit the user preferences. If unable to do so, it notifies the user. This method is a promising way for users to directly manipulate the expected results, shortening the gulf of execution [Hutchins 1995] and facilitating semantic interaction [Endert et al. 2012].

However, the matrix is updated after each iteration; due to changes in decision boundaries, some grids in the confusion matrix change as well (either increasing or decreasing). Although ManiMatrix supports locking the direction of changes, it is not trivial to predict how a given change will impact the results. In addition, it is unclear how often locking the grids leads to unsolvable constraints. Furthermore, like EnsembleMatrix, it is difficult to keep track of changes in ManiMatrix; after a few iterations, users may get lost.

One other example system in this direction is *Dis-Function*, proposed by Brown et al. [Brown et al. 2012]. Dis-Function builds customized distance functions for high-dimensional datasets. The Dis-Function interface visualizes data distributions for users on a 2D scatter plot based on multi-dimensional scaling (MDS). Users manipulate the distance between points by dragging them closer to each other. The system automatically updates the weights in the distance function and updates the interface. To test the distance function, the authors use the derived distance function for a classification problem using the k-nearest neighbor (k-NN) algorithm. Their results show improved accuracy, though the original distance function already yielded about 90% accuracy. While Dis-Function provides an intuitive way to manipulate the distance function, the author notes that sometimes the changes may lead to a completely different layout.

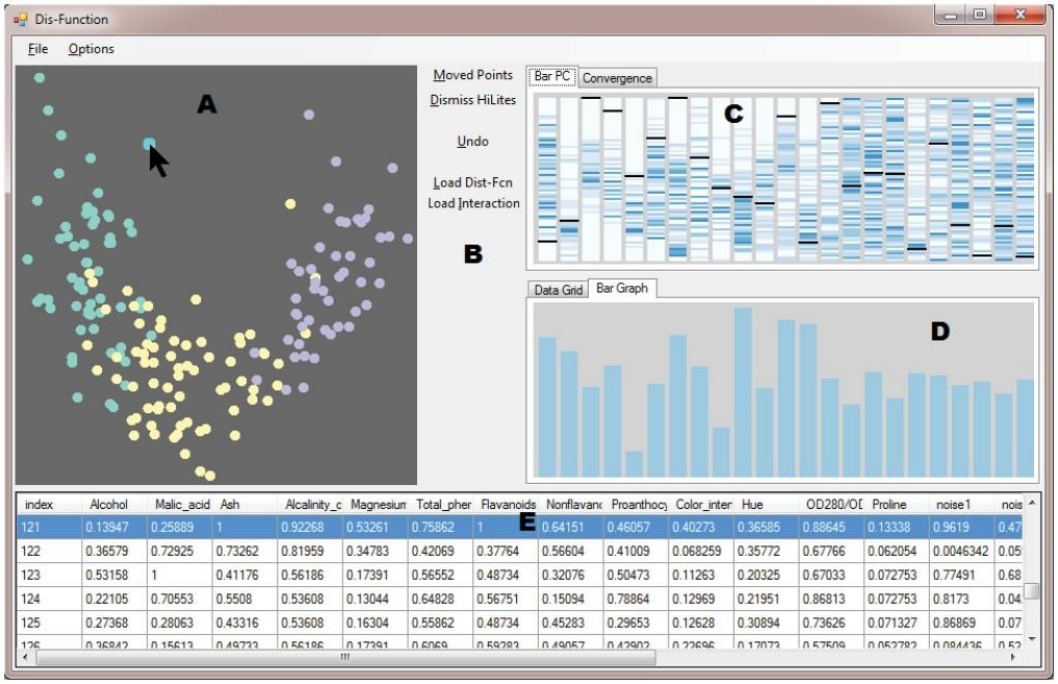

Figure 2.6: Dis-Function

Some work in this direction focuses on providing an intuitive way to manipulate clustering. For instance, in Kim's work on the interactive Bayesian case model [Kim 2015], users specify data points as clustering prototypes as well as important features (i.e., subspaces). In the use case for programming education, Kim shows that users could select a representative example (i.e., a piece of code) as a prototype for clustering and highlight key tokens of the prototype as its subspace. While example-based reasoning has been shown to be intuitive, this approach is similar to the previous two in that it is too difficult to gauge how human input (i.e., the new prototype and new subspaces) impacts the existing model; it is also hard to track the impact through the interactive process.

These examples benefit from intuitive ways of input, but mapping between user input and the final changes to the models can be difficult. Sometimes, it is hard to tell how a change may

lead to a completely different result. Furthermore, keeping track of the changes remains a challenging task in this direction.

2.1.3 *Fitting into end users' natural workflows*

One other direction worth mentioning is fitting interactive ML into end users' natural workflows. In this direction, the focus is not on building ML models *per se*, but on utilizing user inputs for their end goals to build models that facilitate their tasks.

For instance, Amershi et al.'s *ReGroup* supports the creation of on-demand groups through interactive ML behind the system [Amershi et al. 2012]. During group creation, the system recommends potential members to add, and takes user decisions (whether the recommendations are selected or not) as input in refining the models for recommendation. Their study shows that users create bigger groups faster with the help of interactive ML, although there is no way to verify whether the final results are correct or not. A big challenge for this direction is identifying elements from intermediate steps of end user workflows and using them as input for interactive ML. In addition, it may not be easy to start off with an effective model, and many iterations may be needed to generate useful results. Therefore, it is important to either have some initial models or develop algorithms that can learn proper models within only a few iterations.

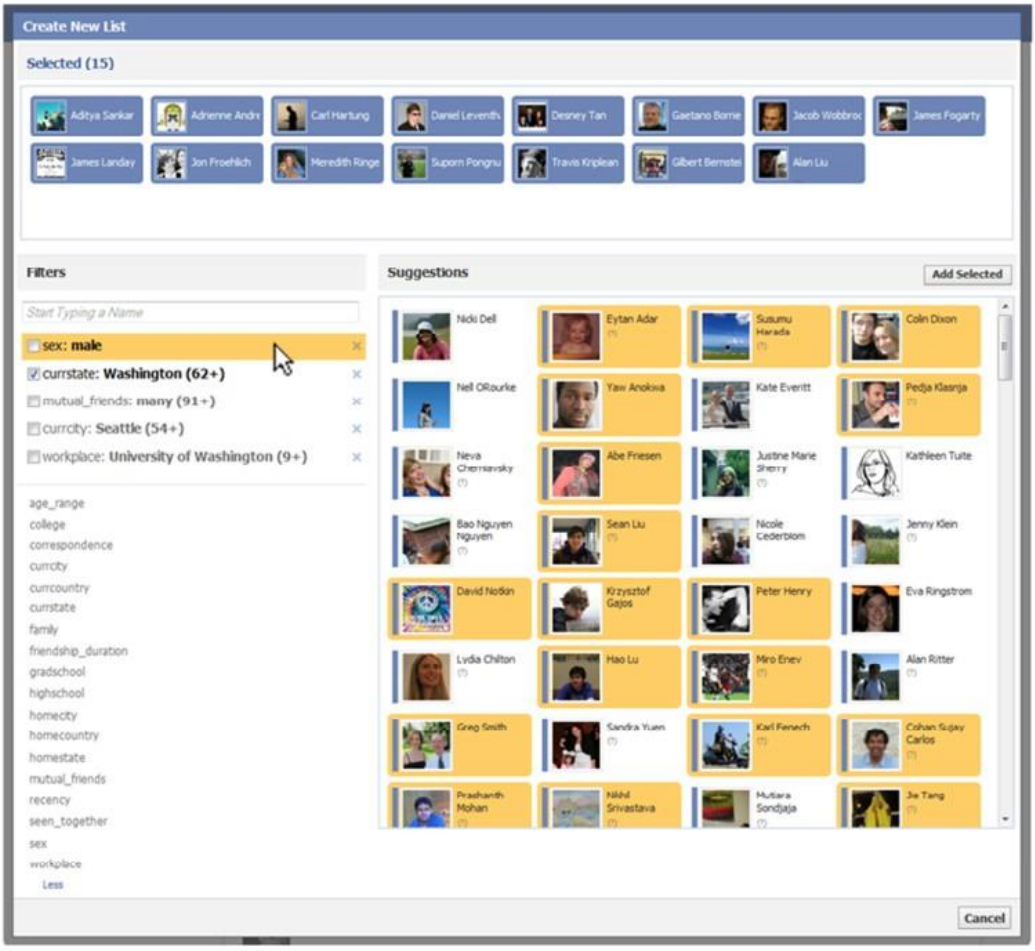

Figure 2.7: ReGroup

2.2 Supporting Machine Learning Model Development

In addition to enabling interactive ML, another thread of supporting AI development is to support ML model development. There are three primary types of work in this thread: (1) using visualization in iML GUIs, (2) using GUIs to support data labeling, featuring, and evaluation analysis, and (3) opening up the black box of ML models. The following subsections will discuss each type respectively.

2.2.1 *Visualization in iML GUIs*

In iML GUIs, visualization is often used as a means to represent ML model status and related numbers (e.g., current accuracy). One key advantage of visualization is that it synthesizes large amounts of information and represents this visually, and as such is usually easier for human perception and cognition to process and/or compare, if used properly. A number of examples mentioned in the previous section use visualization in iML GUIs to inform users of the current status of the model and enable interaction. For instance, in EnsembleMatrix [Talbot et al. 2009], the numbers in the confusion matrix are represented by a color scale (which becomes a heat map). Although visualizing a quantitative value in color results in a loss of granularity (i.e., it is more difficult to distinguish fine differences), the pre-attentive process of human perception can easily capture large color differences [Ware 2012], which fits the purpose of the interface. A similar approach is used in ManiMatrix [Kapoor et al. 2010], where the level of changes is encoded using color opacity with a green/red dichotomy for increases and decreases.

Another common approach (not exclusive to the previous approach) is to use multiple visualizations to represent various aspects of ML models. In interactive ML, where users decide what to do, such an approach is useful to provide an overview from different perspectives. For example, in Ware et al.'s work for interactively building decision trees [Ware et al. 2001], their interface includes a scatter plot showing the distributions of two selected features as well as a set of bar charts showing the distributions of individual variables. A similar idea is used in Dis-Function [Brown et al. 2012] where the results of MDS are shown in a scatter plot along with bar charts for the weights of individual variables on the side. Another example is Mühlbacher and Piringer's interactive framework for building regression models [Mühlbacher and Piringer 2013].

In their work, visualization is used to show the relations between the dependent variable and individual independent variables, as well as the relations between the dependent variable and paired independent variables.

Visualization can be especially helpful when the outputs from ML models are difficult to understand (more discussion is in Section 2.2.3.2). For example, in *iVisClustering*, an interactive visual document clustering system based on topic modeling [Lee et al. 2012], Lee et al. use a node-link diagram to show the relations between documents, and use colors to indicate the cluster numbers annotated by the topic model.

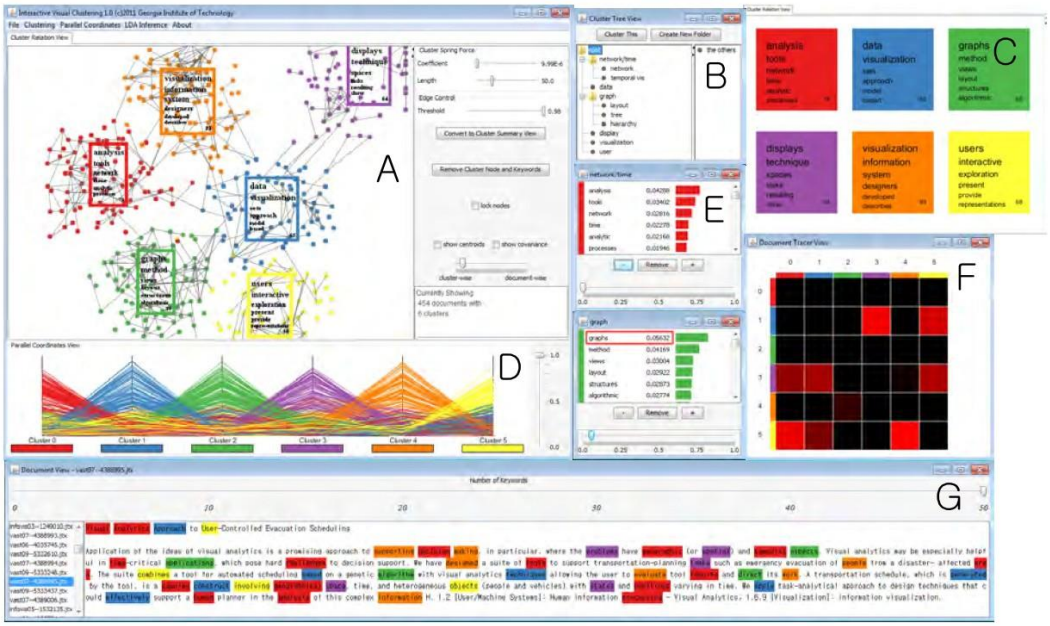

Figure 2.8: iVisClustering

They also show other aspects, such as the top keywords in each cluster and feature distributions, which all use the same color encoding. Furthermore, in documents they highlight the top keywords of each cluster with the corresponding colors. By showing information at

various granularities, users are able to refine the clustering results by changing the top keywords, and remove or merge clusters.

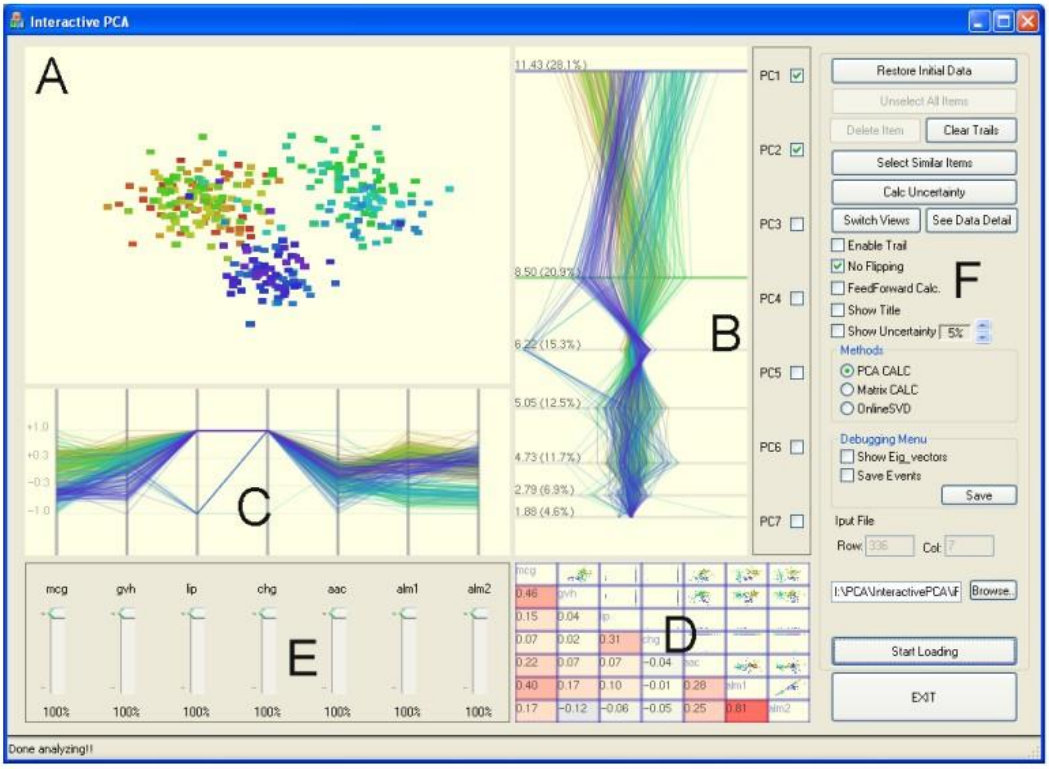

Figure 2.9: iPCA

In another example, iPCA [Jeong et al. 2009] visualizes data point relations and individual feature distributions, and allows users to remove data points and tune weights to update results from principal component analysis (PCA), a dimension reduction method which is generally not very intuitive.

Baobabview, yet another example, is a visualization-based interface to support constructing decision trees interactively. It uses many different representations for various information [Van Den Elzen and van Wijk 2011]: The structure of the decision tree is shown as a tree diagram, where the nodes show the distribution of classes and the links keep track of the divisions of

classes. A streamgraph and a histogram are shown on nodes to show attribute distributions. They also extend the traditional heat map style confusion matrix to encode precision and recall information. The sheer volume of information represented by these complex encodings can be overwhelming to users, since each visual encoding requires a mental effort to remember what it represents. Therefore, even though visualization can be used to examine various aspects of ML, it must be used carefully.

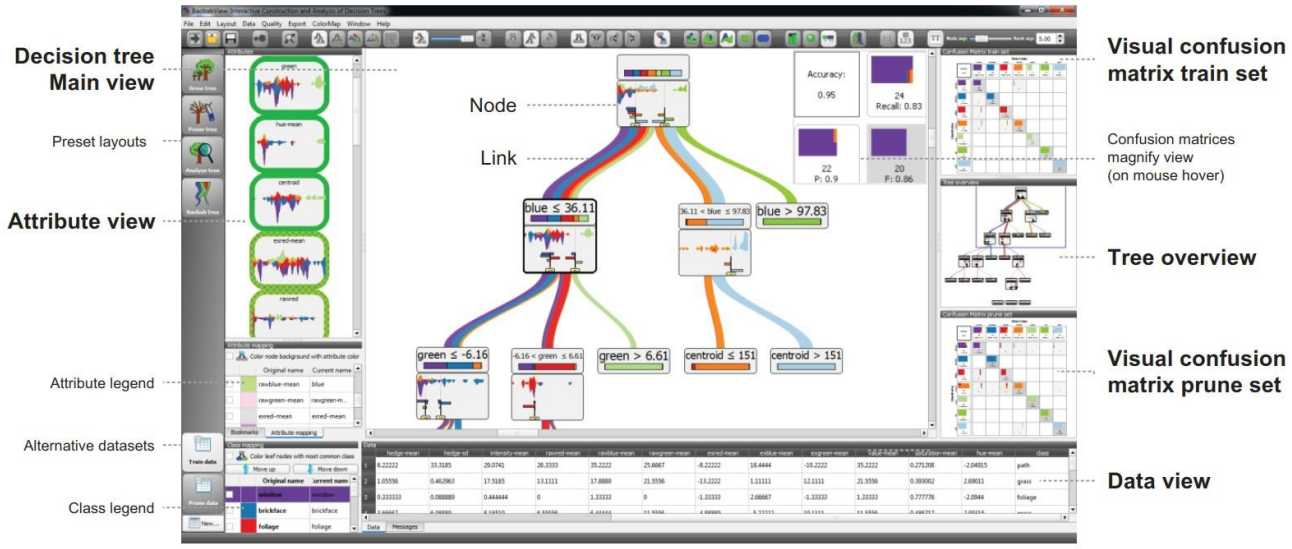

Figure 2.10: BaobabView

2.2.2 *GUIs to support data labeling, featuring, and evaluation analysis*

In addition to using visualization to enable iML, various research directions focus on designing GUIs to support data labeling, featuring, or evaluation analysis. The first focuses on the experience of data labeling, which is a critical step of machine learning. An example work of this is the method proposed by Kulesza et al.: *Structured Labeling [Kulesza et al. 2014]*. They indicate that during data labeling, the labeling concepts in the human mind usually evolve as the

labeler sees more data. Thus, they design a GUI to support what they term *concept evolution* (Figure 2.11).

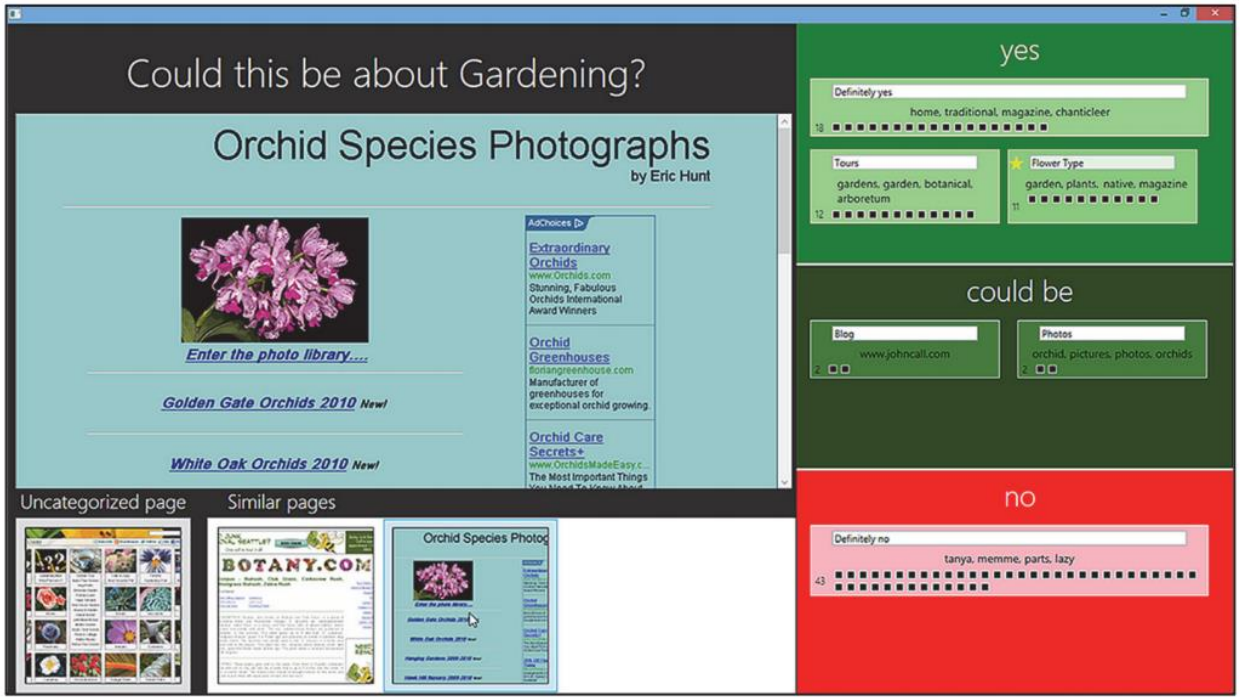

Figure 2.11: Interface to support structured labeling

More recently, Choi et al. proposed a method named *Attentive Interactive Labeling Assistant* (AILA) to use an attention-based deep learning model to highlight what users may need to pay attention to; the labeling results are further used as feedback to the attention model (Figure 2.12) [Choi et al. 2019].

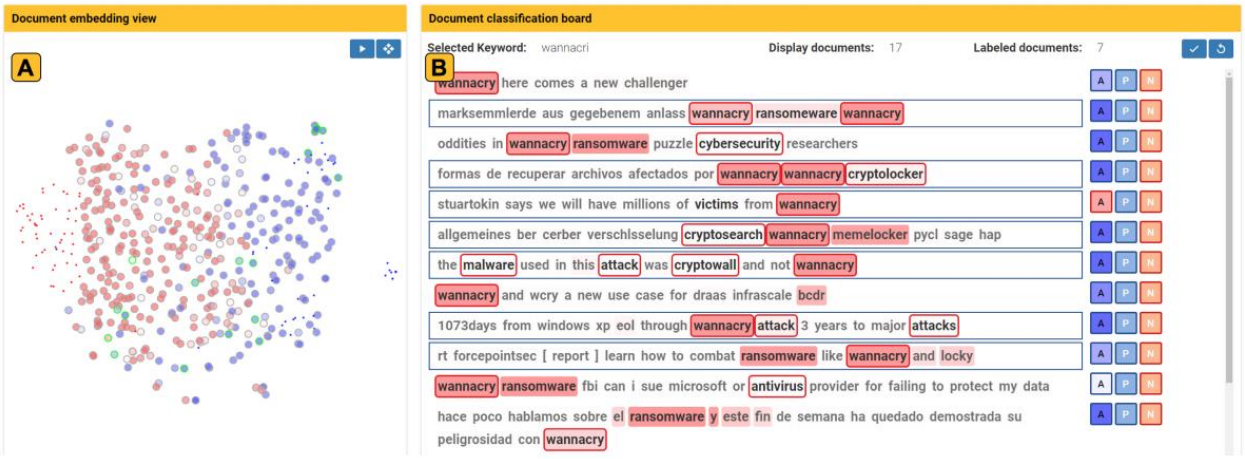

Figure 2.12: Interface of Attentive Interactive Labeling Assistant (AILA)

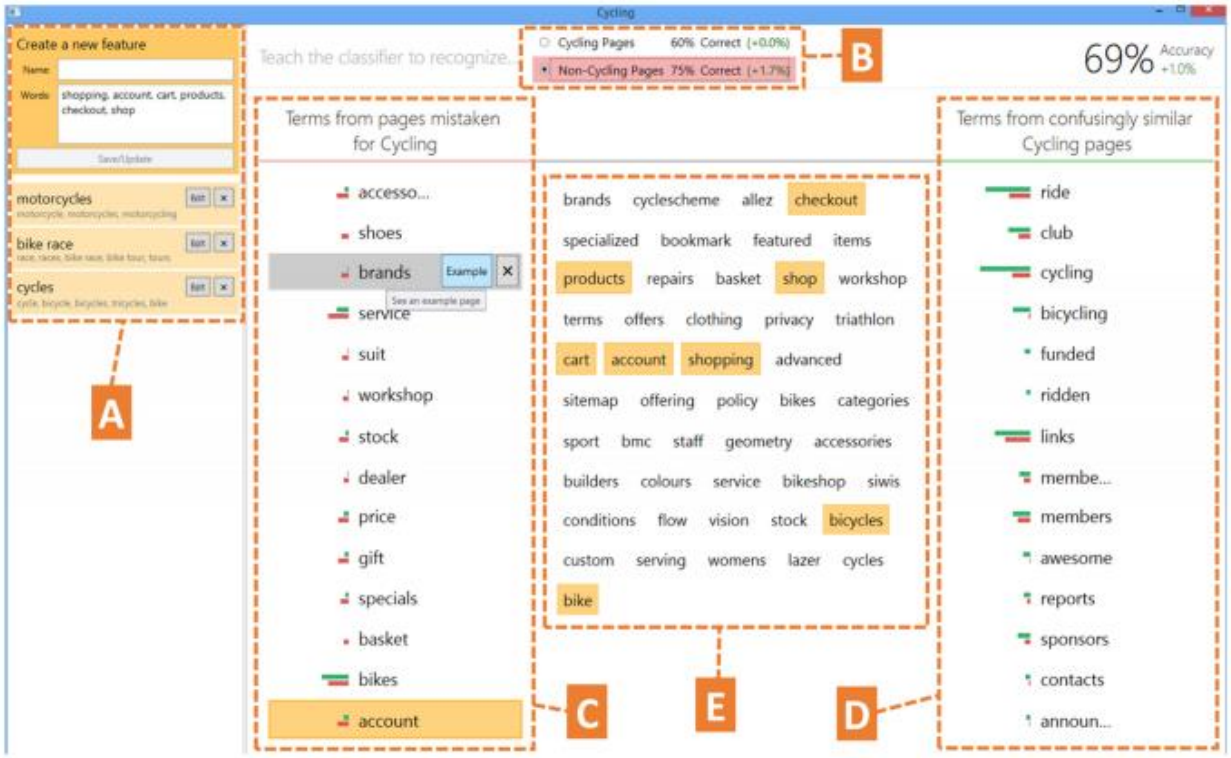

Figure 2.13: Interface of FeatureInsight (Brooks et al. 2015)

In terms of featuring (i.e., generating features to represent target classes), one representative work is Brooks et al.'s *FeatureInsight*, which aims to facilitate feature ideation for text classification [Brooks et al. 2015]. In the GUI design, they focus on showing how features are associated with training errors and enable interactive exploration of examples and modification of the features (Figure 2.13). With the interactive loop, users can come up with new features and ensure feature quality.

In addition to facilitating data labeling and featuring, various works focus on leveraging visualization to enhance the experience of analyzing evaluation results. This thread of work does not focus on the details within the model, but rather focuses on the intermediate and final results in evaluation as the goal is to facilitate analysis. For example, Amershi et al. design *ModelTracker* to show how predictions on a dataset from a binary classifier change over time (Figure 2.14) [Amershi et al. 2015]. Ren et al. further extend the design to multi-class classifiers (both tools refer to this as *performance analysis*) [Ren et al. 2016]. Both tools focus on highlighting classifier errors and allow users to interactively inspect how things change over classification iterations (Figure 2.15). Similarly, Alsallakh et al. present a visual analytics tool that makes sense of the different types of results (true positive, true negative, false positive, false negative) and their relations to the features (Figure 2.16) [Alsallakh et al. 2014]. In their more recent work, Alsallakh et al. present another visual analytics tool, named *Blocks*, to analyze classification errors based on a hierarchy [Alsallakh et al. 2017]. Namely, they indicate that classification errors are usually related (e.g., in image recognition, the same class of animals may be wrong due to a lack of labeled data in the corresponding group); presenting the hierarchy can help developers find patterns of errors and resolve them together (Figure 2.17).

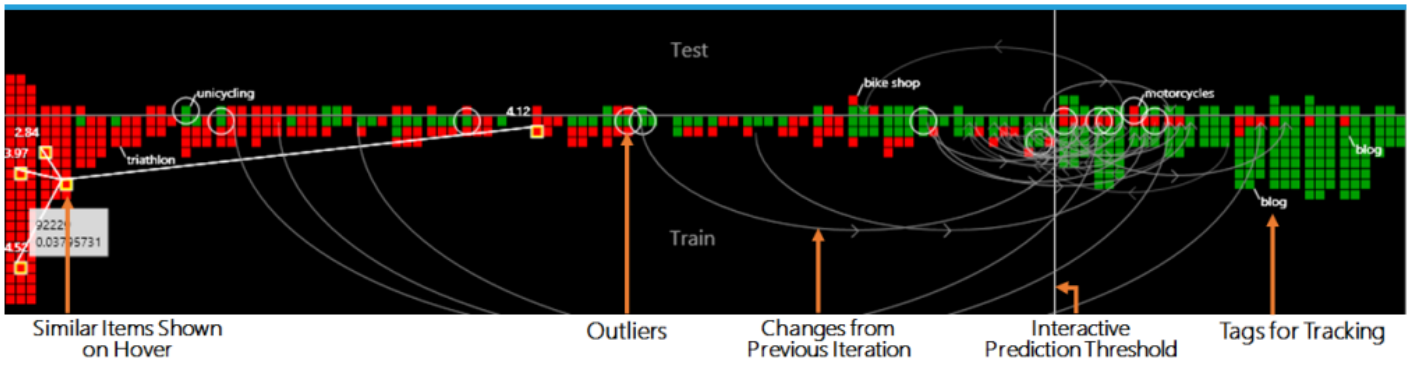

Figure 2.14: Screenshot of *ModelTracker*'s interface

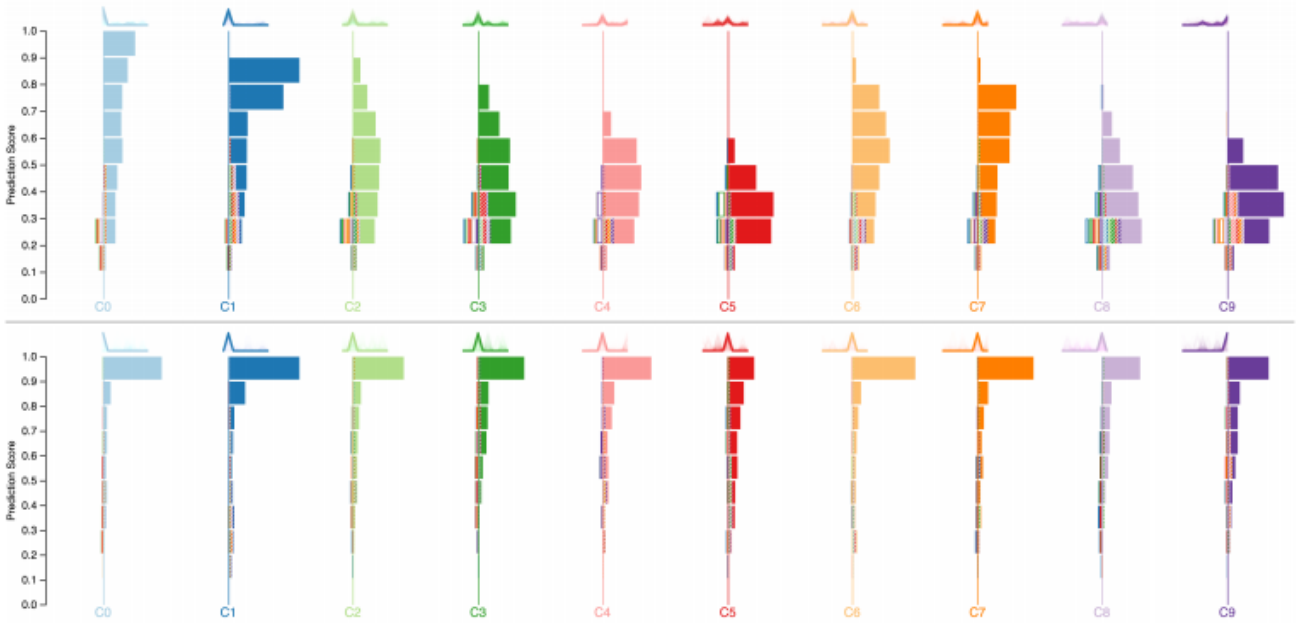

Figure 2.15: Interface of *Squares* for performance analysis on multi-class classifiers

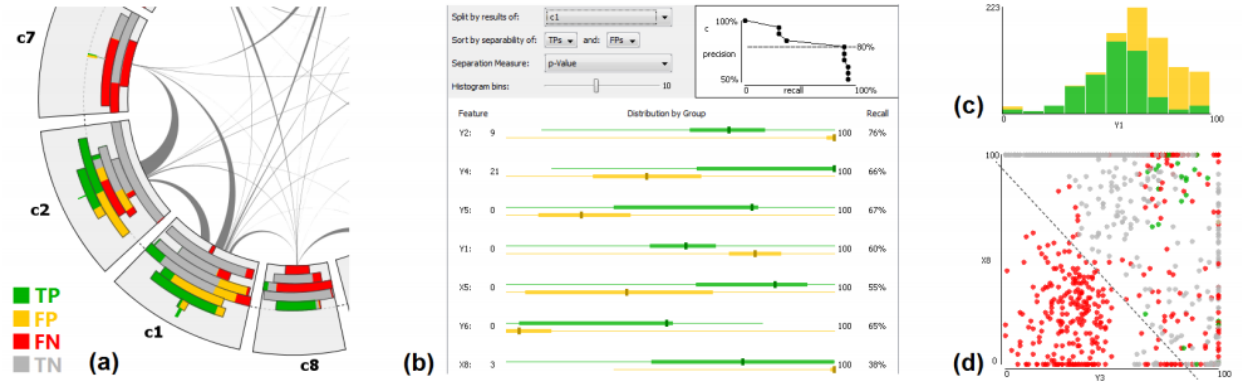

Figure 2.16: Screenshots of Alsallakh et al.'s VA tool interface for analyzing classification errors

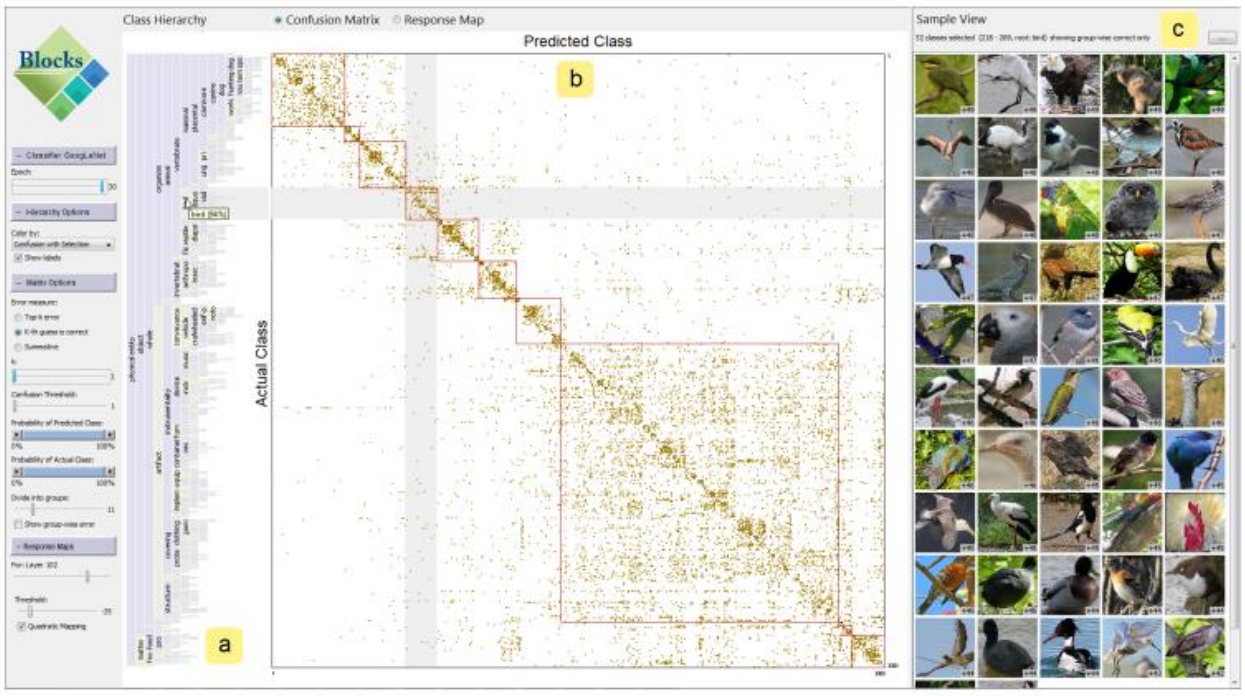

Figure 2.17: Interface of *Blocks* for analyzing classifier errors based on class hierarchy

These tools all attempt to help users to come up with insights on the results based on the relations between inputs (data and corresponding features and labels) and outputs (prediction

scores/predicted labels). In the next subsection, a different approach is described, which focuses on opening up the "black box" of ML models.

2.2.3 *Opening up the black box to enable analysis*

One key challenge of current ML development is to understand the models and how changes made to the models impact final results. This issue is related to the interpretability of ML, namely, how humans can understand the internal states, outputs, and reasoning behind the decisions of an ML model. Previous work has shown that displaying explanations for model reasoning can increase user trust [Pu and Chen 2006]. This section will first focus on work concerning interpretable models, and then discuss the role of visualization in this topic.

2.2.3.1 Interpretable ML models

According to Kim [Kim 2015], work on interpretable models can be roughly divided into three directions: sparse linear methods, discretization methods, and case-based methods. An example of the first direction is a recent paper by Ribeiro et al. [Ribeiro et al. 2016]. In their paper, Ribeiro and colleagues propose *LIME*, a method for explaining classification results for a given data point by sampling nearby data points to generate explanations. This method is further extended by Lundberg and Lee [Lundberg and Lee 2017], who consider feature importance through Shapley values from game theory and coin their method *SHAP (SHapley Additive exPlanations)*. The sparse linear approach assumes that the behavior of a model can be considered a linear combination of individual components, each of which is interpretable. Furthermore, sparsity can be considered: the fewer components, the more understandable the final results are to humans. Although this approach is helpful (according to their study results), the underlying

assumption may not be valid in all cases. This is similar to the early development of psychology, when people believed the human mind to be the sum of individual parts (a view from Structuralism).

However, Gestalt psychology showed that the mind processes information from a holistic view. Gestalt psychologist Kurt Koffka's famous quote—"The whole is other than the sum of the parts"—indicates the potential fallacy of the sparse linear classifiers/explainers approach, where information from the overview level may be missing [Dewey 2018]. The second direction based on Kim's categorization is discretization methods. The decision tree is the most representative approach in this direction, as its reasoning processes are somewhat more intuitive than most other ML methods. However, some decision trees require more than a thousand splitting points; this may be beyond what humans can understand and manage.

The third direction is case-based methods. This direction takes advantage of the fact that people excel at reasoning based on examples [Newell and Simon 1972]. The examples mentioned above [Amershi et al. 2011, Kim 2015, Pu and Chen 2006] leverage this by showing the most representative examples in their classification or clustering results.

One important assumption of this approach is that the membership of groups is to some extent interpretable. That is, given the members of a group (a class or a cluster) and the most representative example of the group, one can tell how other members in the group relate to the example. For cases like images, membership may be more obvious and easier to understand, but for cases like text, it may become more challenging. In addition, sometimes a certain number of data points are required to make sense of the membership, which complicates the application of this approach to cases with limited numbers of data points.

Other work approaches interpretability in yet different ways. For instance, instead of measuring interpretability of models through mathematical formulations, Chang et al. propose two human tasks—*word intrusion* and *topic intrusion*—to evaluate how interpretable a topic model is [Chang et al. 2009]. This human judgment-based approach can supplement computational methods, although it may be expensive to use iteratively. Moreover, this approach does not provide an indication of how to improve the interpretability.

Another example is related to the recent trend of deep learning, which is known to be difficult to understand [Zeiler and Fergus 2014]. Some attempts have been made to use visualization to help people understand what is happening in a neural network (e.g., Figure 2.18 [Yosinski et al. 2015], Figure 2.19 [Zeiler and Fergus 2014]), while others try to describe the expressivity of a neural network. For instance, Raghu et al. propose three metrics—*transitions*, *activation patterns*, and *dichotomies*—and connect these metrics to trajectory length [Raghu et al. 2016]. As both approaches capture only some aspects of deep learning, increasing the interpretability of various neural networks remains an open problem.

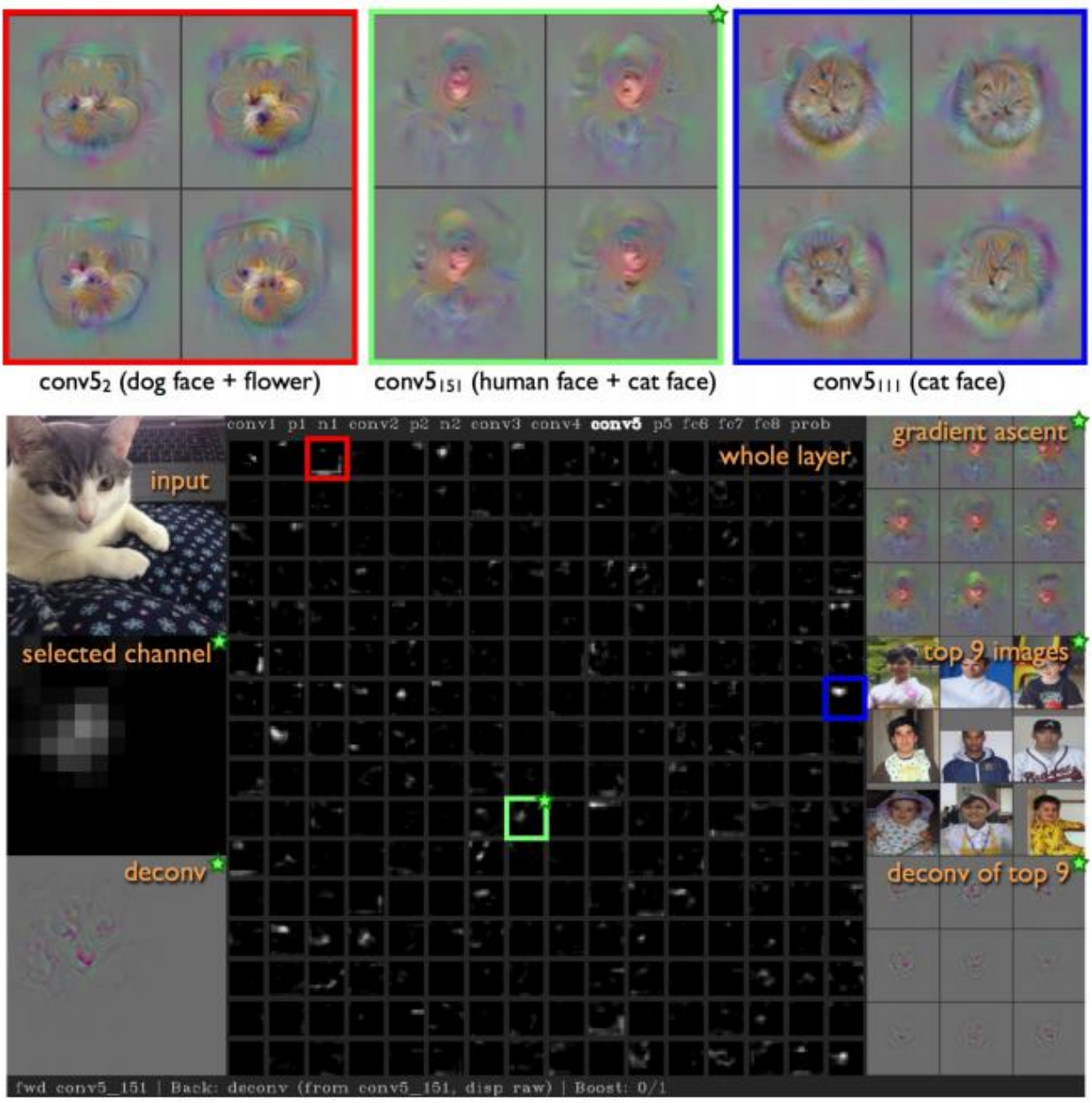

Figure 2.18: Yoshinki et al's visualization method for showing what happens in the intermediate layers of a deep learning model for image recognition

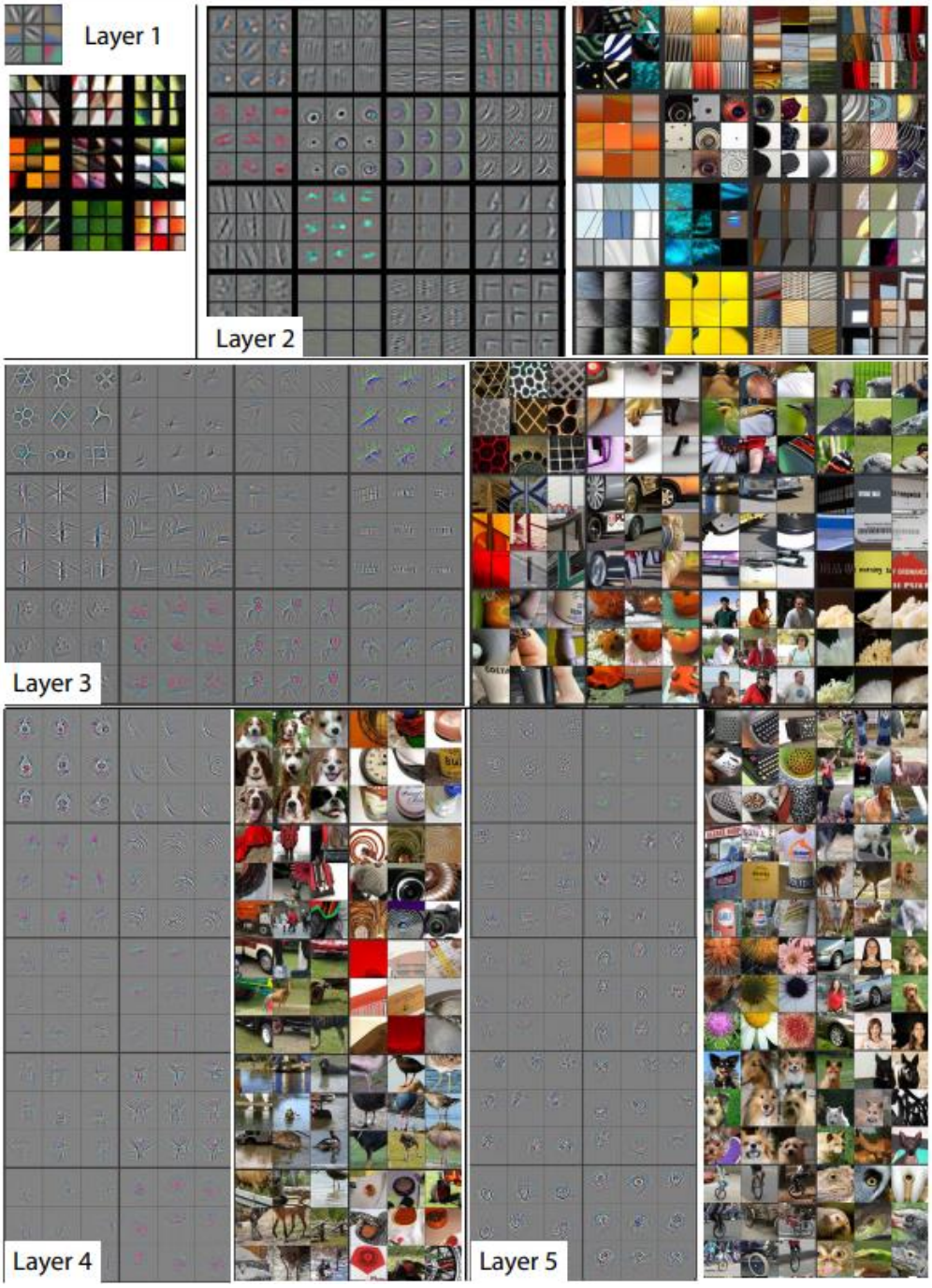

Figure 2.19: Zeiler & Fergus's visualization method to show features learned by their deep learning model

2.2.3.2 Visualization for model interpretation and analysis

Following the trend of interpretable ML models, a variety of visualization work has been proposed to help model developers to explore not only model evaluation results but also intermediate model status. Work in this direction can be divided into two directions: model-agnostic methods and model-specific methods.

Model-agnostic methods focus on using merely inputs, outputs, and features to explain the results. Such work is useful when model internals are not available to the analysis tool. For example, Krause et al. attempt to use local feature relevance to generate instance-level explanations for binary classification results [Krause et al. 2017]. This approach is similar to local explanation methods such as LIME and SHAP. They propose a workflow to leverage the explanation for model diagnosing and design a corresponding interface to facilitate interactive model diagnostics.

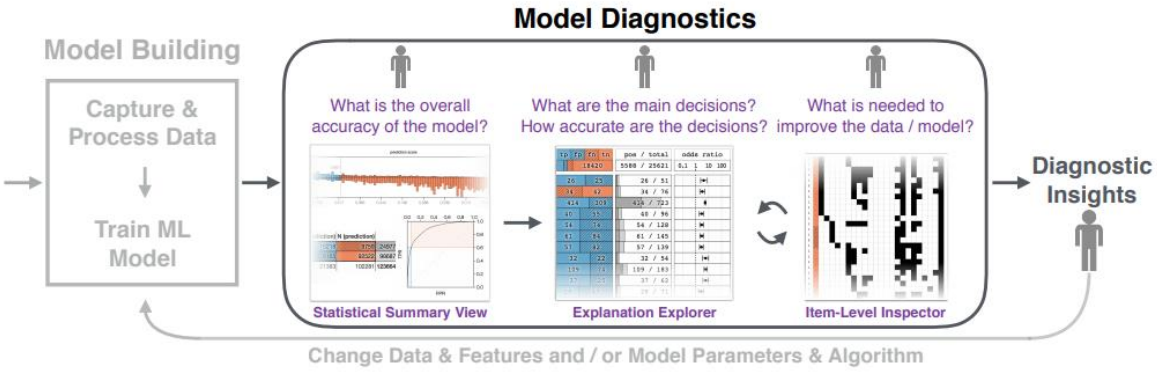

Figure 2.20: Binary classifier model diagnostics workflow with instance-based explanation proposed by Krause et al.

Another work of this type is *Manifold* (Zhang et al. 2018), which focuses on multi-class classification and allows comparison between models [Zhang et al. 2018]. In Manifold, small multiples are used to show pairwise model predictions of each class; colors are used to indicate if

a data point belongs to the corresponding class. In the feature attribution view, each bar indicates the number of instances of that feature within the column class. Users can sort the features by the level of distinct association of the class based on the selected type (C: whole class, G: blue bars/ground truth, N: red bars/misclassified instances), which can help generate insights about representative features.

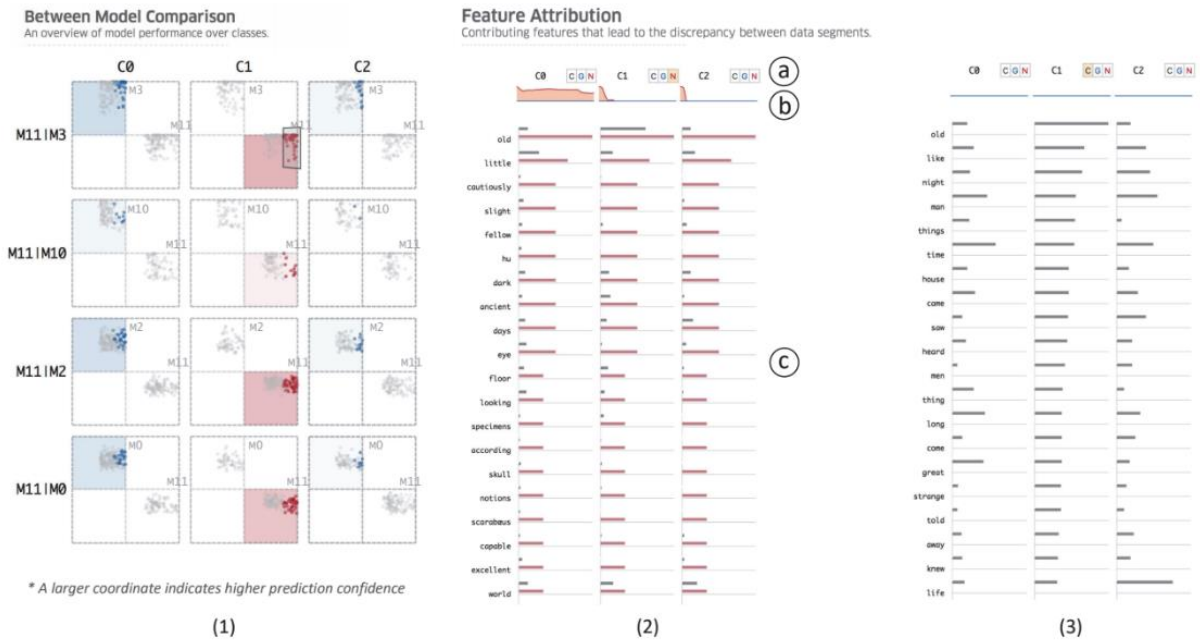

Figure 2.21: Manifold interface

Another type of model-agnostic method can actually be ML-independent, focusing on visually analyzing the relations between data points based on the features/parameters/attributes. For instance, Orban et al. propose a technique called *drag and track* to allow users to directly manipulate data points within a continuous parameter space to play with the points to better understand the relations between them [Orban et al. 2018]. Dowling et al. propose *SIRIUS*, another method to let users not only see the dimension reduction results of the data points but also that of the attributes [Dowling et al. 2018]. Namely, one attribute may be related to another

attribute; knowing the relations between these attributes can help users foresee how data point distribution can affect results. In their interface design, Dowling et al. further allow users to drag points (both data points and attribute points) so that they can see how the change impacts the dimension reduction results from both sides.

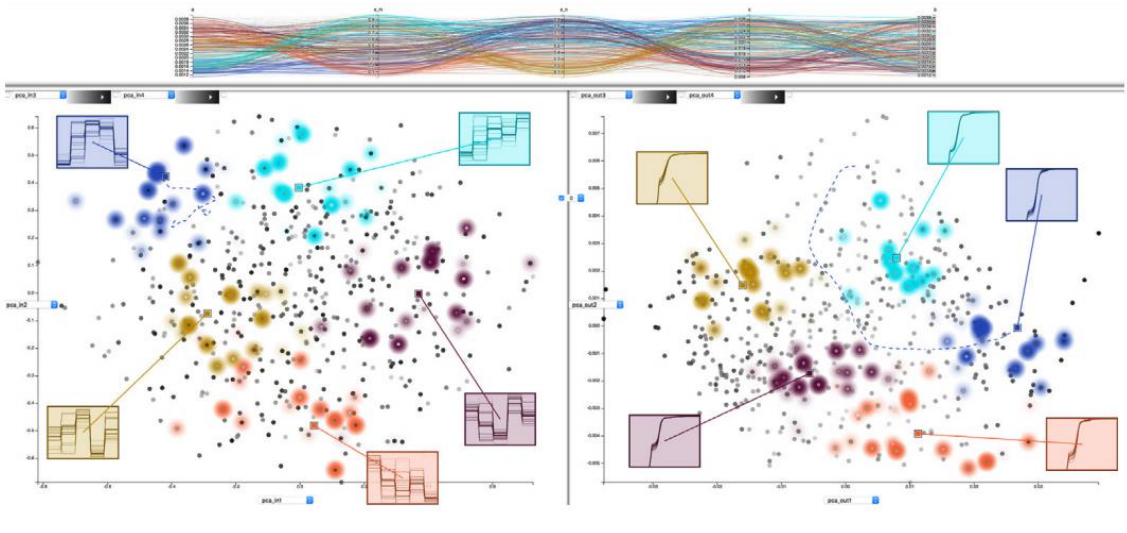

Figure 2.22: Drag and Track interface (Orban et al. 2018)

Figure 2.23: SIRIUS interface (Dowling et al. 2018)

A large amount of work focuses on model-specific methods. One key reason is due to the fact that most ML models, especially deep learning models, are very difficult to understand.

Work in this direction leverages visualization to help model developers understand and explore what happens in the model.

Work on visualization to support interpreting specific models can be divided into two classes, based on whether it focuses on traditional ML models or deep learning models. For traditional ML models, a majority of work examines unsupervised learning results such as topic modeling and clustering. The need for interpretation of such models comes from the nature of unsupervised training. Namely, the grouping of data is based on intrinsic data attributes, which may or may not have a real meaning. Therefore, it is important to interpret the results and verify that the resulting clusters/topics are meaningful, as well as to allow users to compare and make modifications when necessary.

For example, Kwon et al. propose *Clustervision*, a visual analytics tool to allow users to compare different clustering setups and provide a summary of different clustering metrics as a glyph [Kwon et al. 2017]. Users can further examine the details of each clustering result to decide which clustering setups to use.

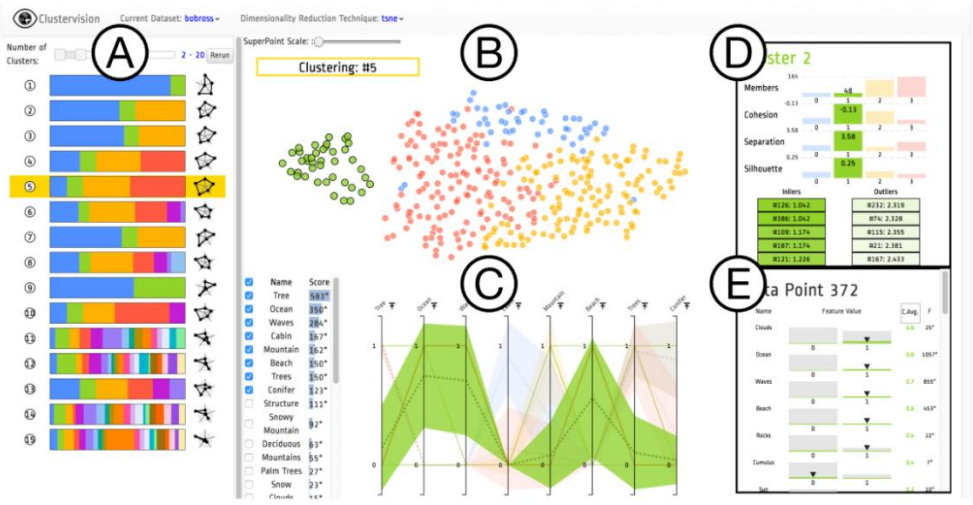

Figure 2.24: Clustervision interface

El-Assady et al. focus on building visual analytics for interactive topic modeling [El-Assady et al. 2017, El-Assady et al. 2018]. In their 2017 work, they propose an interface that allows comparison of two topic modeling parameter setups. The focus is to let users examine which top words are associated with each topic and the corresponding categories of each word. The tool also enables reinforcement learning to take into account user feedback when generating the next iteration results so that users can observe the progress of the topic model.

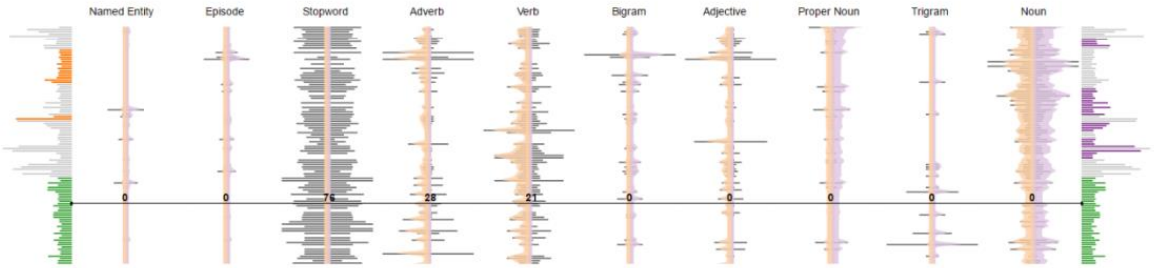

Figure 2.25: El-Assady et al.'s work on progressive learning of topic modeling parameters

In 2018, they proposed another design to let users see and compare the impact of potential changes. In this new design, instead of focusing on the top word association distribution, they put more emphasis on the topic structure and enable users to steer the model execution: the tool suggests possible modifications to the topic model which users can preview to determine whether they want to accept it. With such a design concept, users can better understand the model, which promotes trust.

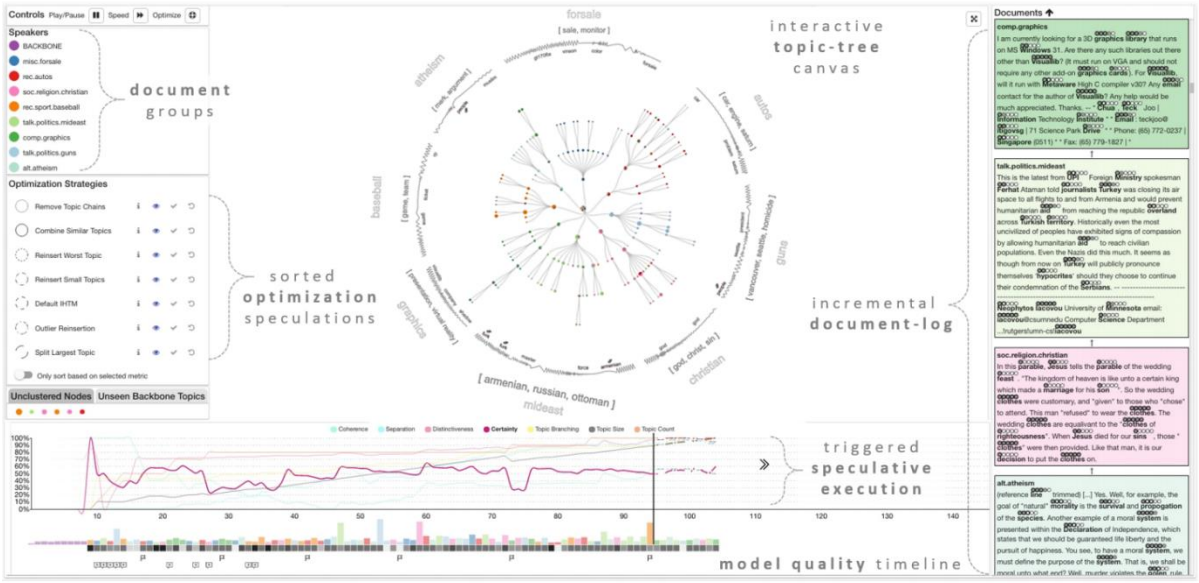

Figure 2.26: Interface design of El-Assady et al.'s 2018 work to support user steering in topic modeling

In addition to unsupervised ML methods, there has also been work on interpreting supervised ML models. For instance, *iForest* [Zhao et al. 2018] attempts to interpret random forests via a visual analytics approach. Random (decision) forests are an ensemble method of decision trees. Whereas decision trees are usually considered an intrinsically interpretable method, decision forests add complexity, making the model more difficult to interpret. Therefore, in iForest, Zhao et al. uses a visual representation and interaction to help users understand the collective behavior of the decision trees. The design of iForest includes a data overview, a feature view, and a decision path view. The data overview projects all data and the forest prediction decision onto a 2D plane. The feature view shows the partial dependence plot (PDP) of each feature, feature distribution, and decision split point distribution. In the data path view, the project view is used again, but this time each point is a decision path (from a tree root to a tree leaf). This view shows the different prediction paths of one data point and their relations

to each other. The goal is to help users understand how different decision paths can lead to different prediction results.

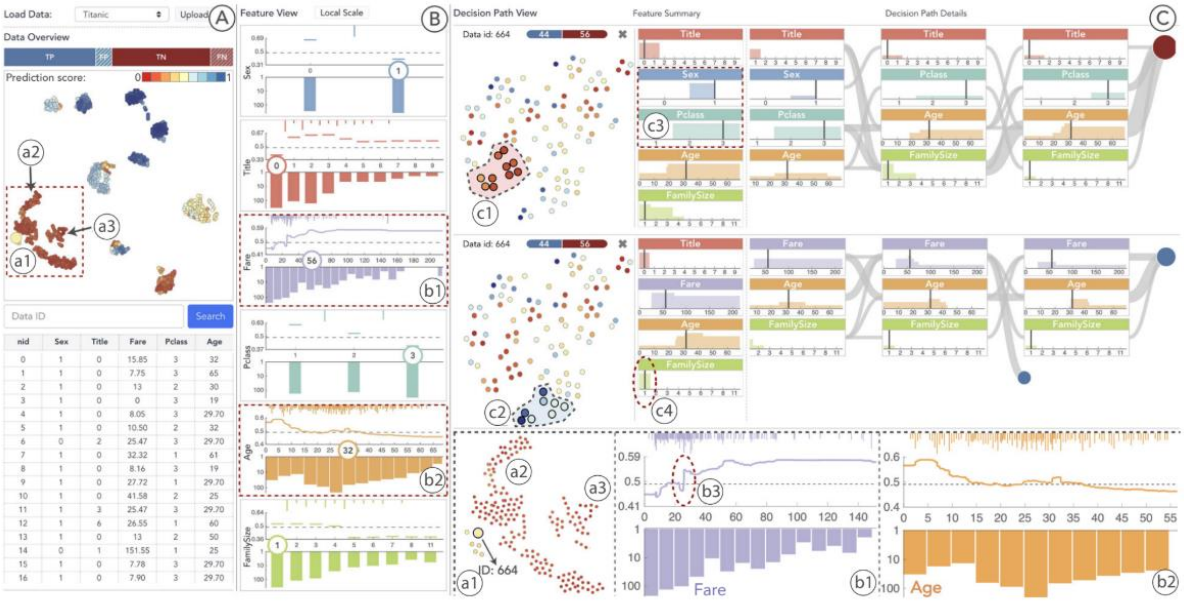

Figure 2.27: iForest interface

Another recent work on interpreting traditional ML model is Dingen et al.'s *RegressionExplorer* [Dingen et al. 2018]. In their interface design, Dingen et al. emphasize the need to compare different multivariate models as well as conduct subgroup analysis. That is, there are different ways to consider all the variates; each subgroup can have different performance. By allowing users to split the population, it becomes easier to interpret how each model has different results on each subgroup, which facilitates selection of the final model.

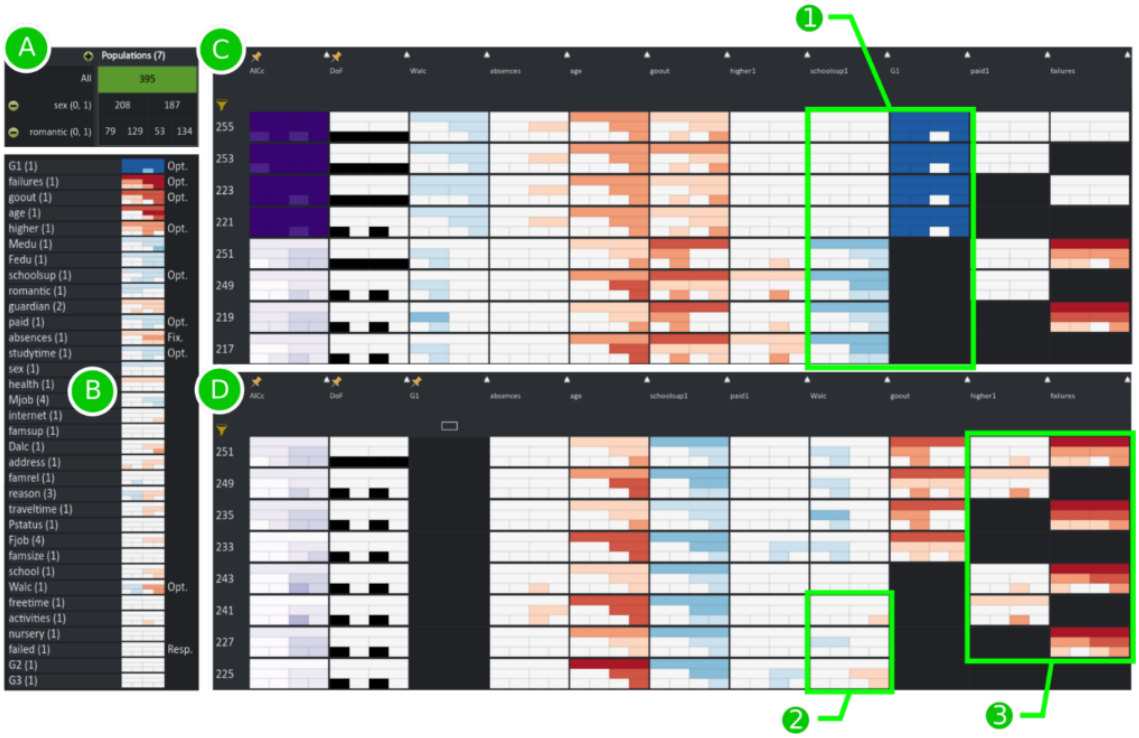

Figure 2.28: Subgroup analysis view of RegressionExplorer

As deep learning becomes more prevalent in ML research, more attention is being given to designing visualization tools to help interpret deep learning models; it is still a keen topic of research. A variety of tools have been designed for various types of deep learning models and/or components.

Some work focuses on embeddings, which is a technique often used to convert discrete data into vectors and preserve certain properties (e.g., similarity distance). Embeddings can help reduce the size of the input dimension used in the deep learning network. However, embeddings are not easy to interpret either. Thus, some have attempted to develop visual analytics tools to help model developers understand and explore these embeddings. For instance, *EmbeddingVis* [Li et al. 2018] allows comparisons between different embedding methods, allowing users to see how a data point distributes differently in various methods.

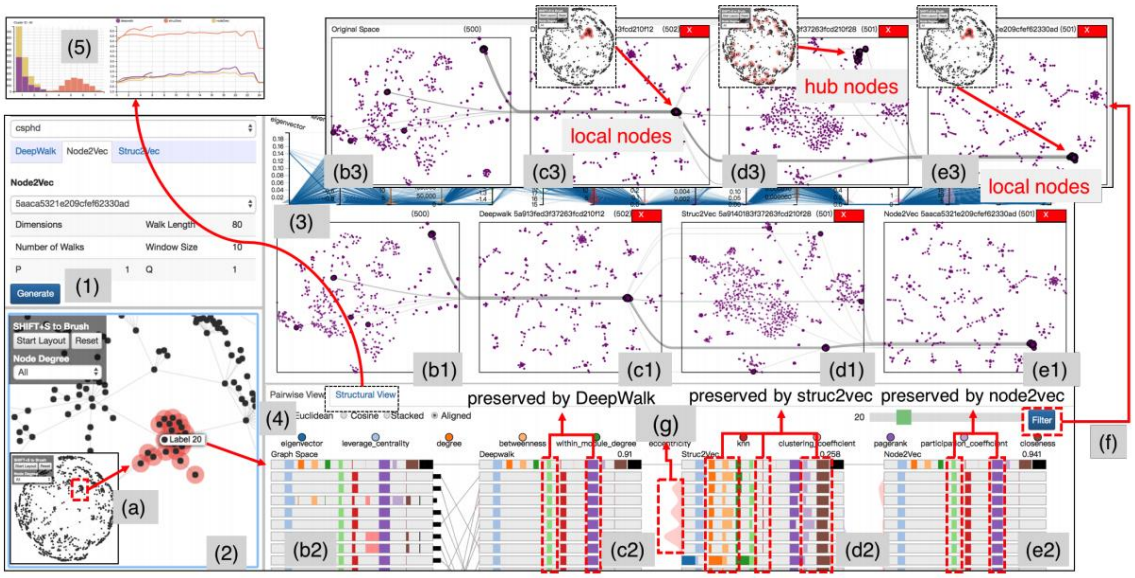

Figure 2.29: EmbeddingVis interface

In contrast to general embedding visualization, Liu et al. support visual exploration of the semantic relations of word embedding results [Liu et al. 2017], also leveraging the t-SNE embedding algorithm (a dimensionality reduction algorithm widely used for visualization).

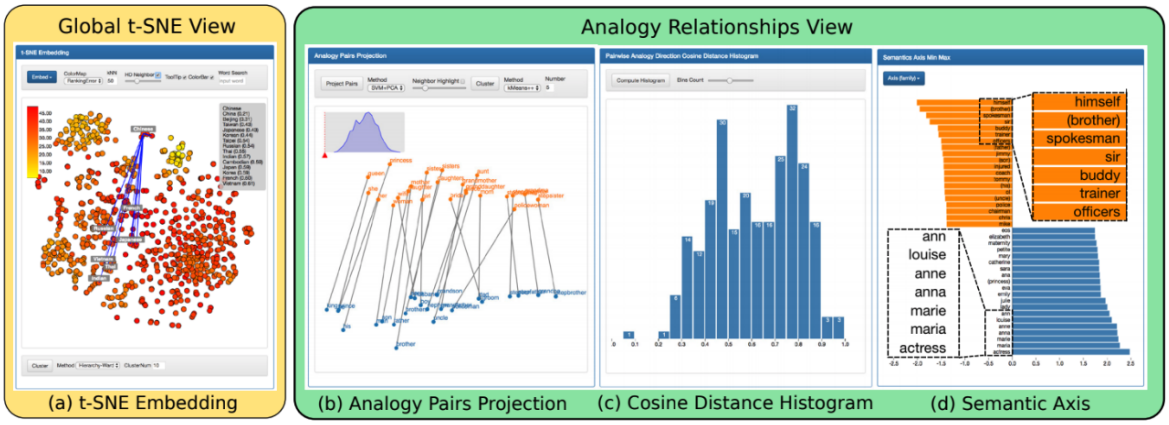

Figure. Screenshot of exploration interface of semantic relations for word embedding results (Liu et al. 2018)

Another common design target when visualizing deep learning models is visualizing the network structure/architecture. Such visualization contains a diagram indicating the neurons in each layer. Some work in this direction shows how data flows over the network. For instance, Wongsuphasawat et al. design a dataflow graph for TensorFlow (a widely used ML framework from Google) that provides the model developer with an overview of the network structure and the corresponding dataflow [Wongsuphasawat et al. 2017].

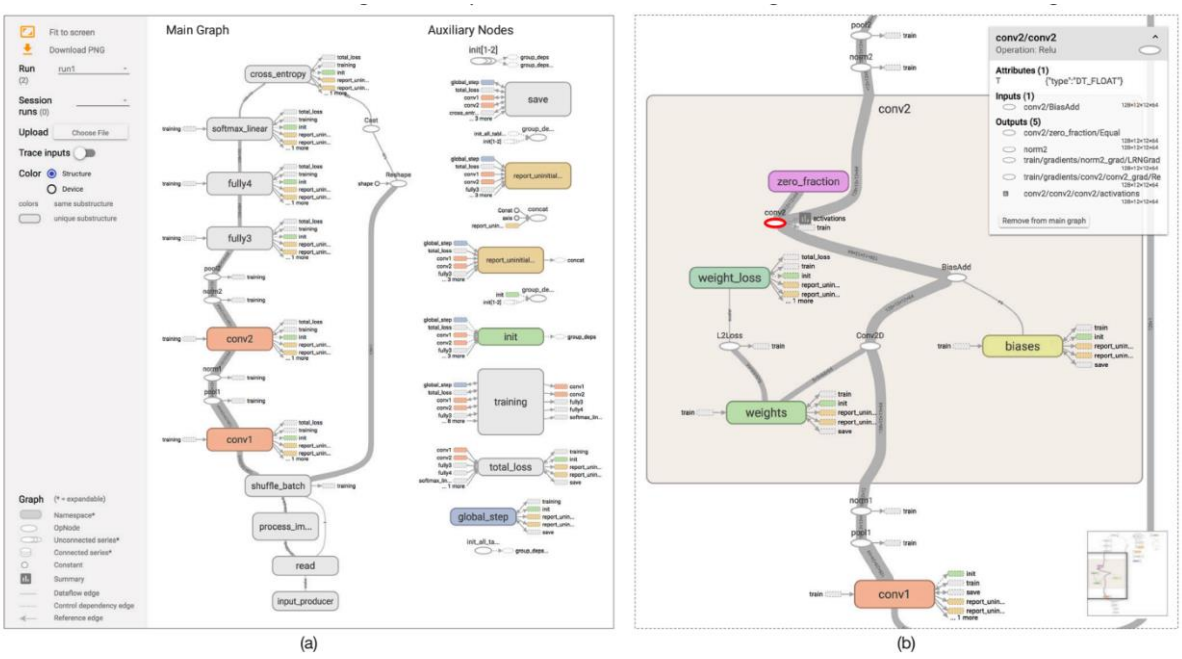

Figure 2.30: Dataflow graph in TensorFlow designed by Wongsuphasawat et al. (2017)

Other work shows how each neuron is activated over the network structure. For instance, Kahng et al. propose *ActiVis* to allow users to examine how neurons are activated [Kahng et al. 2017]. Users explore nodes of the network structure to see neuron activation. Users also select specific data points, or sets of data points based on customized criteria and observe how each neuron is activated.

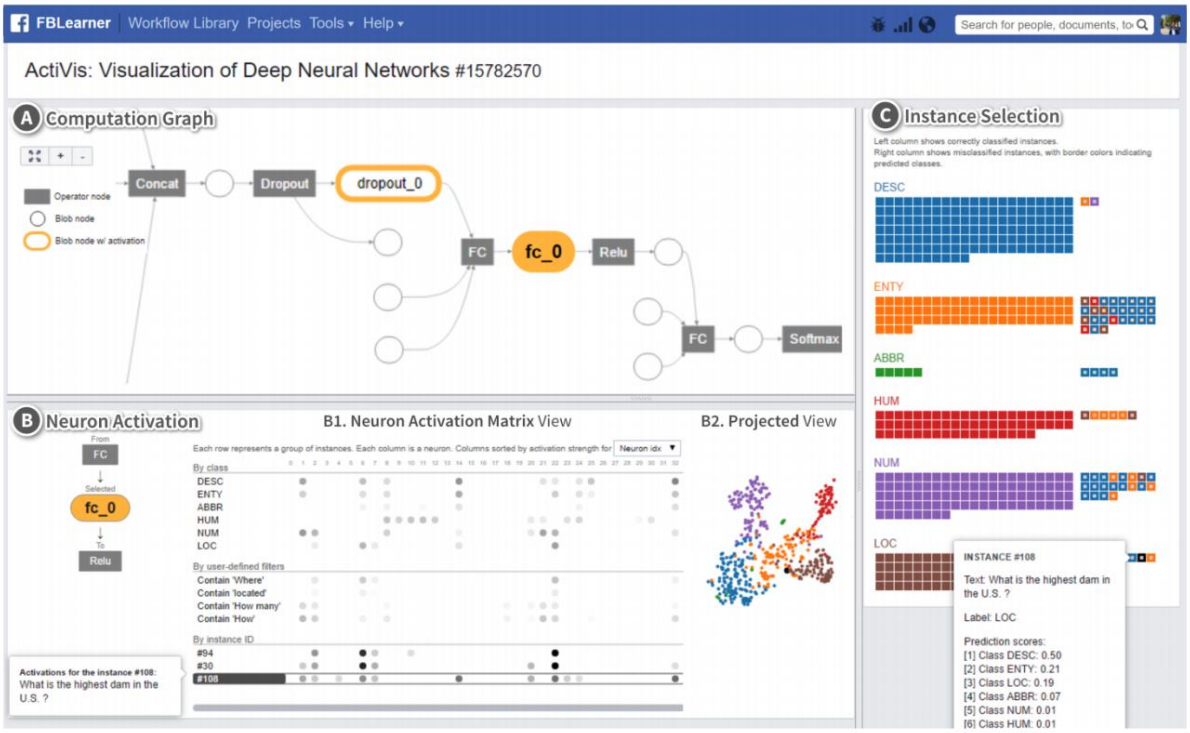

Figure 2.31: ActiVis interface by Kahng et al. (2018)

Also focusing on representing information over network structure, Liu et al. (2016) propose a hierarchical rectangle packing algorithm and a matrix reordering algorithm to visualize a convolutional neural network (CNN) along with the intermediate results while preserving basic network layer structure [Liu et al. 2016]. The key purposes of the two algorithms are to reduce the visual clutter at the edges and provide condensed information in each of the matrices (the squares at each layer). Users view and navigate on this overview visualization and further investigate the details in each neuron, to better analyze the relations between neurons.

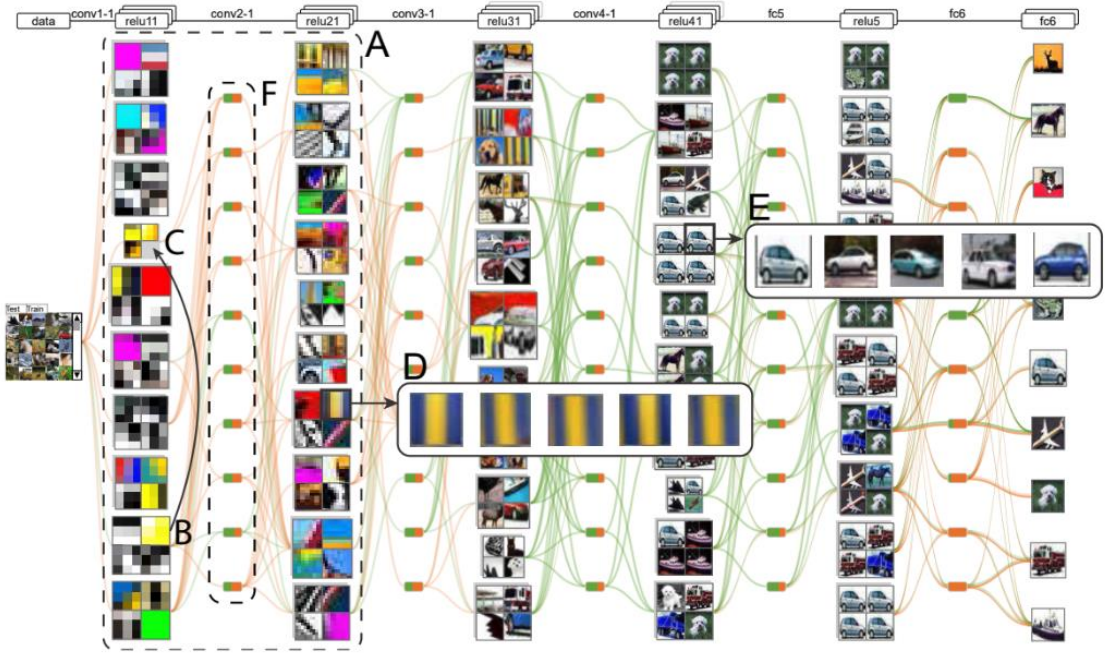

Figure 2.32: Example view of Liu et al.'s work (2016) for better analysis of CNNs

In their series of work on visualizing CNN models, Liu et al. use visualization to support the analysis of adversarial examples [Liu et al. 2018]. In this work, they still preserve the network structure, but instead of the matrix view, they use a simple diagram representation of the network. Users interact with each node to view how a neuron is activated and examine the feature map to observe why the network fails to recognize the adversarial example of the target class.

Another way of visualizing NN models is to put all neurons/filters together without directly showing the network structure. With this method one can discover patterns based on neuron status/attributions without being constrained by the network structure. Example work in this direction includes Ming et al.'s work on visualizing hidden memories of recurrent neural networks (RNNs) [Ming et al. 2017] and Pezzotti et al.'s *DeepEyes*, a visual analytics tool [Pezzotti et al. 2017].

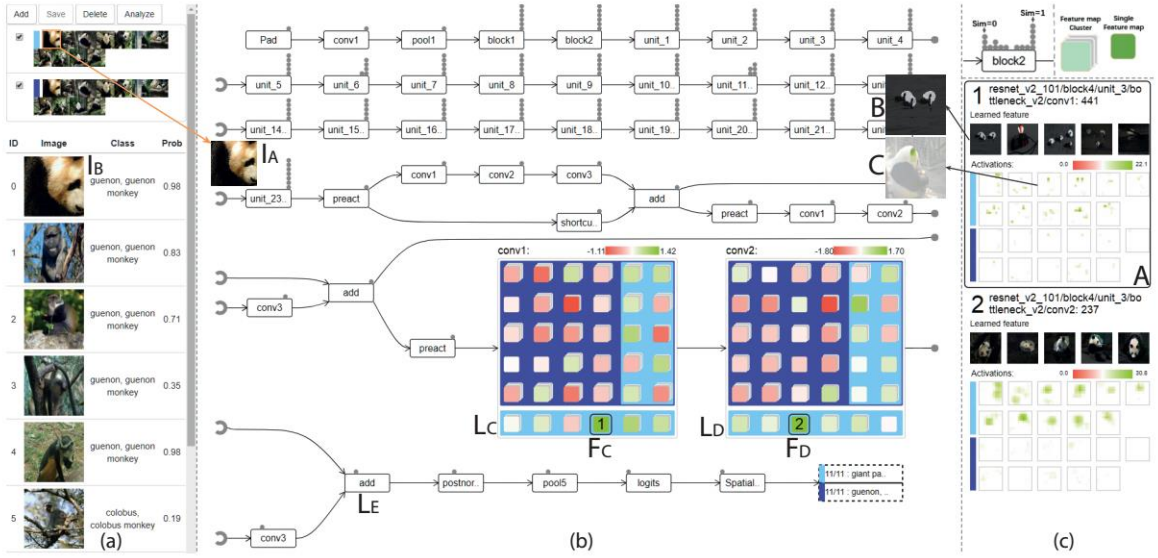

Figure 2.33: Liu et al.'s interface design for analyzing adversarial examples in CNNs

Ming et al. cluster and visualize hidden memory units in an RNN that has similar patterns at the sentence level [Ming et al. 2017]. To be more specific, they use a co-clustering algorithm to cluster both hidden memory units and words to aggregate information from both sides, and then come up with a visual representation for the hidden memory unit cluster and visualize the words with word clouds. By such a representation, users see how a given sentence is reflected in the hidden memory clusters, and what the representative words of these clusters are.

Also, to break down the structure while organizing information, Pezzotti et al.'s *DeepEyes* shows activations on filters and the relations between filters [Pezzotti et al. 2017]. The idea is to help users identify useless filters so that they can modify the network based on the current results. Users also observe how filter activations change over training iterations to learn about the network behavior.

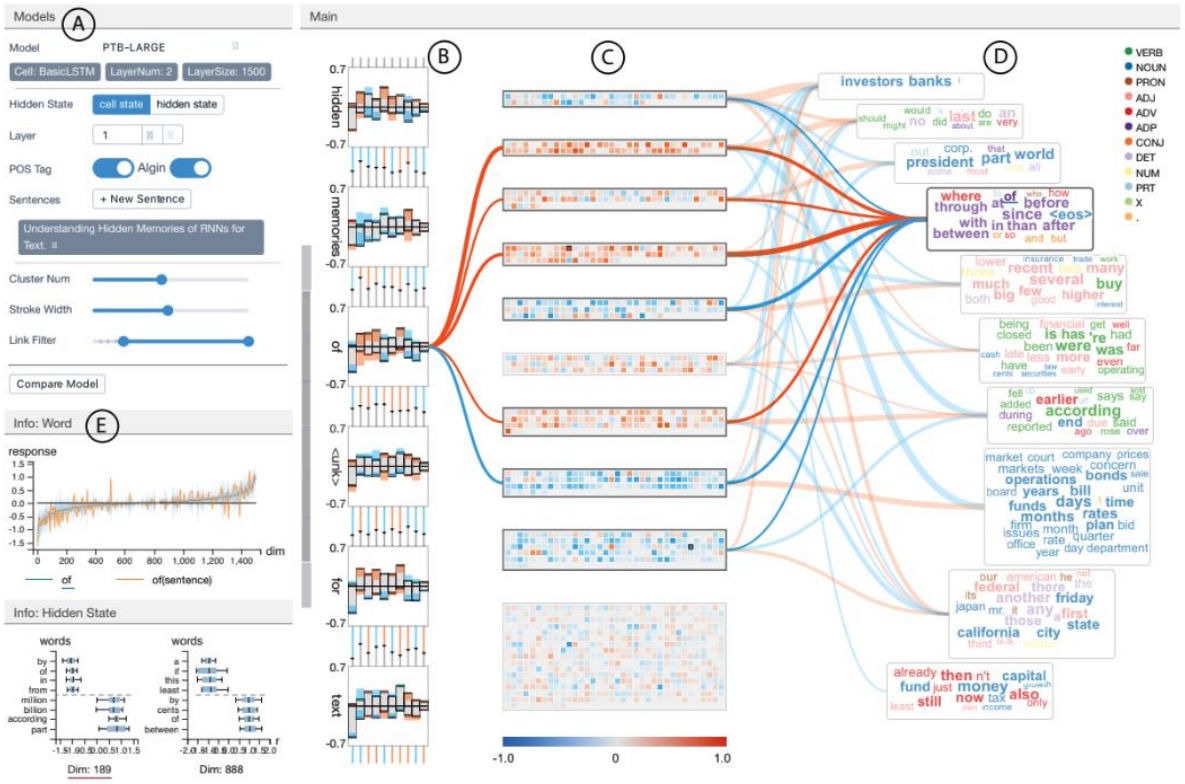

Figure 2.34: Ming et al.'s interface to help analyze hidden memory units in RNNs

Figure 2.35: DeepEye interface (Pezzotti et al. 2018)

Another direction in visualizing deep learning networks is sentence-level information, which is specific to NLP deep learning models. For example, Strobelt et al. propose *LSTMVis*, which also helps users understand hidden state dynamics [Strobelt et al. 2017]. In their interface design, users observe how each part of the sentence activates the hidden units. In the selection view, users can even define a range to instruct the tool to find other sentences with similar activation patterns. Users compare the results of different activation patterns.

Figure 2.36: LSTMVis interface, designed by Strobelt et al. (2017)

In their further work, Strobelt et al. extend the scope with *Seq2SeqVis*, which focuses on sequence-to-sequence (Seq2Seq) models, which are often used in fields like machine translation. This type of model often has an encoder and a decoder model. Thus, Seq2SeqVis supports switching between two models [Strobelt et al. 2018]. In addition, it also allows users to dynamically modify the mapping between input/output words and attention. Furthermore, the

tool shows the nearest neighbors of the current model state. This facilitates the process of analyzing Seq2Seq model results, yielding a better understanding between the encoder/decoder model states.

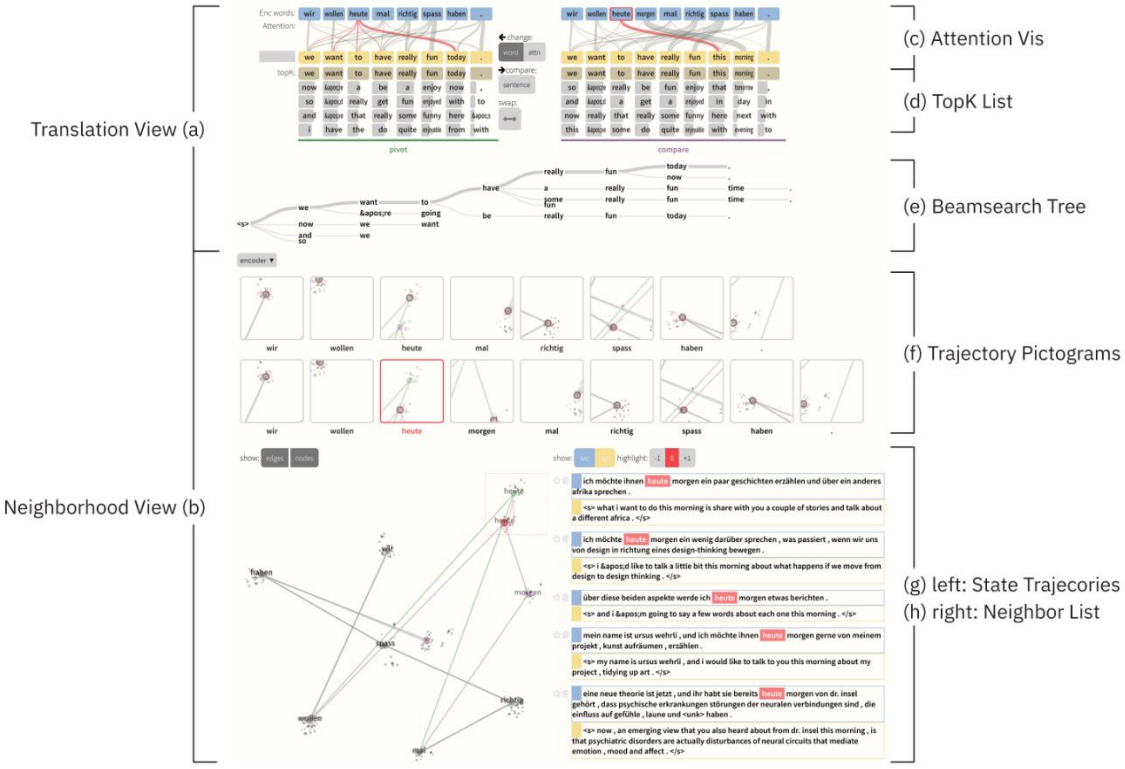

Figure 2.37: Seq2SeqVis interface by Strobelt et al. (2018)

Another type of deep learning model that has been visualized is Deep Q-Network, which is a type of deep reinforcement learning network. Wang et al. propose *DQNViz*, a visual analytics tool for examining a DQN model and its behavior in *Atari 2600* games [Wang et al. 2018]. The primary target of analysis is the behavior over time and the corresponding environment at the time. To determine whether the actions the agent takes make sense, users examine the rewards and actions of the game agent and view the segment that the agent really sees.

Figure 2.38: DQNViz interface by Wang et al. (2018)

2.3 Summary

In summary, past research using visualization includes tools to enable iML and tools to support ML development. For tools that enable iML, there is a paucity of work focusing on helping AI developers look for classifier errors in unlabeled datasets, which is the focus of AnchorViz (Chapter 4). For supporting AI development, none of the work reviewed here is designed for AI systems (most focus on ML models). In addition, the focus of previous work is not on analyzing and comparing results from various experiments. Thus, in Chapter 3, with the design of QSAnglyzer we aim to address the challenges of analyzing evaluation results of QA systems, which is a type of AI system. The following two chapters describe the detailed studies for designing the above two tools. Furthermore, little existing literature focuses on providing an overview of AI development workflows which is critical to surface challenges and opportunities

for designing VA tools. Thus, in Chapter 5, we conduct an interview study to propose a framework for describing AI development workflows and highlighting issues in current practices.

Chapter 3. QSAnglyzer: Visual Analytics for Evaluation Analysis

3.1 Introduction

As mentioned in the introduction chapter, building an AI system is still a challenging task. In this chapter, we will focus on the design of a visual analytics tool for a particular type of AI system: a Question Answering (QA) System. A QA system is a type of complex AI system that gained recognition in public when *Watson*, IBM's QA system appeared on the *Jeopardy!* game show [Ferrucci et al. 2010] in 2011. But QA has been a subject of study for four decades now. Numerous QA systems have been proposed to answer short factoid questions [Brill et al. 2002, Ferrucci et al. 2010, Ko et al. 2007], standardized tests in geometry [Seo et al. 2014], and mathematics [Kushman et al. 2014]. Some QA systems are designed to answer multiple-choice questions (e.g., [Clark et al. 2016, Wang et al. 2014]), whereas others are designed for direct-answer questions (e.g., [Voorhees 1999, Zheng 2002]).

Typical QA systems can be roughly divided into two types: information retrieval (IR)-based and knowledge-based systems [Jurafsky and Martin 2009]. IR-based QA systems rely on large sets of unstructured corpora and use IR techniques to search for sentences that contain words found in the questions. In contrast, knowledge-based QA systems are built upon structured knowledge (e.g., lists of tuples for relations between semantically typed entities) for reasoning. The knowledge base is usually smaller and more organized than IR corpora, and the knowledge can be chained to infer unseen facts.

In addition, in complex AI systems, ensemble methods are often used to combine results from a set of weaker AI systems to generate the final outputs. Likewise, a QA system may comprise a set of *solvers* that answer questions, and use ensemble methods to produce the

system's final answers. In other words, a solver can be a QA system itself, but it is usually used together with other solvers in ensemble methods for producing final answers. In this chapter, we will focus on supporting evaluation analysis on these individual solvers.

Since there are many ways to design QA systems and to combine them, analyzing the evaluation results to design what to use and how to combine them becomes challenging. For instance, when changes are made in one component, it is not easy for AI developers to discern detailed behavior or behavior change patterns from either final metrics or single point results.

One way to aid this investigation of evaluations is to introduce categorization. Showing finer-grained patterns may provide deeper insight than showing overall metrics. The challenge is that researchers do not know a priori the best way to categorize evaluations. In addition, the best insight to improve the system may require iterative filtering on various types of categories. The complexity of such exploratory queries can further impose burdens on AI developers.

In this chapter, we tackle the challenge of evaluation analysis in the domain of QA systems. We worked closely with QA developers to identify their goals and tasks in analyzing QA evaluations, and derived a set of design rationales that support their workflows. These in-depth studies with QA developers led to the novel approach of *prismatic analysis*. This approach examines data from multiple angles, where each angle shows a different way to divide the input data into categories. Categories are then measured by aggregate metrics for further investigation. With this approach, users carry out various types of comparison scenarios, including between-category, between-angle, and between-evaluation comparison. In addition, as the approach works with data subsets as well, QA developers can conduct analysis even after filtering.

Although multiple angles of categories can enable more diverse pattern discovery, they can also be overwhelming to researchers. To support the prismatic analysis of QA evaluations, we design and implement a visual analytics (VA) tool called the *Question Space Anglyzer* (QSAnglyzer). In QSAnglyzer, questions in evaluations constitute a high-dimensional space in which each question is a point. The question space can be divided into categories based on several angles, for instance topics and question types. The aggregate metrics for each category that are chosen based on the design rationales include accuracy, the number of questions, and accuracy variance across evaluations. QSAnglyzer visualizes these angles and aggregate metrics of categories using color, height, and order. The QSAnglyzer design enables QA developers to examine and compare evaluations from various angles individually and collectively. Furthermore, QA developers filter questions based on any angle by clicking, thus constructing complex queries visually. With controlled experiments and expert reviews, we confirm that the tool helps users conduct analysis more quickly ($p < 0.01$) and with greater accuracy ($p < 0.05$), and also generates insight in a short amount of time. The major contributions of our work are as follows:

- We identify the goals and tasks of QA developers in analyzing evaluations to derive design rationales. We highlight the need to further study the workflows of complex AI system development.
- We propose prismatic analysis, a novel approach for multi-angle categorization and shared aggregate metrics for category comparison and connection, which enables finer-grained pattern discovery.
- We design and implement QSAnglyzer, a visual analytics system for conducting prismatic analysis on QA system evaluations, and validate its effectiveness. We discuss how prismatic analysis and QSAnglyzer scaffold evaluation analysis, and we provide

directions for future research to extend the tool and apply prismatic analysis to other AI domains.

This chapter is my internship work at Allen Institute for Artificial Intelligence (AI2) under supervision of my mentor, Been Kim. The original paper was published in IEEE VIS, Visual Analytics Science and Technology (VAST) track in 2017 [Chen and Kim 2017]. Been Kim and I conducted the user studies and wrote the original paper together. I contributed the design ideas, iterating with target users, implementing, evaluating, and analyzing QSAnglyzer.

In this chapter, we address our thesis statement by studying and designing QSAnglyzer, a VA tool for analyzing the evaluation results of QA systems. We seek to understand how to design VA tools for AI development, and verify the design ideas through HCD approaches, which includes a formative study, iterative design with QA developers, and user testing with expert QA developers and non-QA-expert AI developers.

3.2 Related Work

3.2.1 *Visual analytics (VA) for multi-experiment result analysis*

Conducting multi-experiment result analysis, including comparison and exploration, is one of the most common uses of visualization in scientific domains. For instance, Nocke et al. propose a series of visualization views for climate science simulation output comparison [Nocke et al. 2007]. Wilson and Potter focus on visualizing ensemble simulation data for weather forecasting and support side-by-side comparisons [Wilson and Potter 2009]. Malik et al. utilize visualization to inspect sets of 3D X-ray computed tomography images where device parameters varied when taking the images [Malik et al. 2010]. Although such applications are common in scientific domains, they do not directly apply to QA system design for three reasons. First, most

above-mentioned work focuses on simulation data from the same models with different parameter settings, which is not the case in QA system design. Tuning parameters is only one type of change that QA developers make to their solvers. Oftentimes the changes are more complex, such as adding more knowledge, changing language processing algorithms, and so on. Furthermore, there is typically more than one type of solver, and they all work very differently from each other; thus there is no intuitive way to visualize the parameter spaces across these heterogeneous solvers. Such unique characteristics of QA systems compose a different design space for visualization from the typical comparative visualization used in scientific domains.

3.2.2 *VA for inspecting computational models*

Inspection of computational models such as ML models is a growing topic in the field of human-computer interaction (HCI) and visualization (VIS) research. Two key goals of this direction are to better understand complex black-box models, and to reduce the effort required to construct and tune models. The former goal aims to make it easier for model designers to pin down potential issues with the complex computational process, whereas the latter provides clues for improvements to models. For instance, Smilkov and Carter demonstrate an interactive visualization that shows the process of neural networks (NN) [Smilkov et al. 2017] so that model designers can see how a NN model progresses over time. Various work in classifier weight tuning [Kapoor et al. 2012, Talbot et al. 2009] or feature ideation [Brooks et al. 2015] also demonstrates how visualization can simplify efforts to construct a better model. Although QA and other complex AI systems also contain computational models, enhancing the performance of such complex systems requires more than focusing on specific models. Therefore, we focus on

the workflow level of the development process, and specifically focus on building analytics that support evaluation analysis.

3.2.3 *VA for inspecting computational models, multi-view learning, and multiple clustering solutions*

Prismatic analysis can be related to multi-view learning and multiple clustering solutions. Multi-view learning considers a machine learning problem from multiple views (e.g., multiple sources or different feature subsets) since the target learning problem cannot be described using a single view [Xu et al. 2013, Zhao et al. 2017]. Multiple clustering solutions focus on using multiple ways to cluster data points [Muller et al. 2012], and have been used in subspace analysis of high-dimensional data space [Tatu et al. 2012]. Like multi-view learning, prismatic analysis also emphasizes the need to consider multiple views (here referred to as *angles*), but it does not focus on learning problems per se. In addition, multi-view learning does not require defining categories in each view. In contrast, for multiple clustering solutions, the goal is to define categories (i.e., clusters) in many different ways: this is similar to the idea of prismatic analysis. However, prismatic analysis can be positioned as the next step after finding multiple clustering solutions. The results from multiple clustering solutions can be fed into prismatic analysis. Moreover, multiple clustering solutions focus on finding categories, whereas prismatic analysis emphasizes comparisons and making connections between categories. Thus, defining aggregate metrics is also a critical step of prismatic analysis, which is not a part of multiple clustering solutions. In Section 3.4, we describe prismatic analysis in detail.

3.3 User Requirement Analysis

In this section, we describe user requirements and design rationales derived through in-depth user studies with QA developers.

3.3.1 *Identifying user requirements*

To identify user requirements in analyzing QA system evaluations, we worked with a research team with about 20 people who built a QA system at a research institute in the USA for six months. The formative study happened mostly in the first month of the project.

During the formative study, we informally interacted with the team on a daily basis, and we conducted a series of interviews with five QA developers who were primary solver developers. All of the interviewees held PhD degrees and had been working full-time on the project as research scientists for more than a year; their experience in related fields ranged from 10 years to 33 years.

The QA system consisted of multiple solvers for answering standardized science exam questions. The system used an ensemble method to combine answers from the solvers, and returned a final answer for each question. For simplicity, we hereafter use ‘solvers’ to refer to both solvers and the QA system, as the QA system could be considered an ensemble solver that outputs answers based on all solvers’ answers.

3.3.1.1 QA evaluation overview

To develop and test the QA system, the QA developers ran solvers on a set of evaluation questions. When we worked with the QA developers, their set contained about 1,200 multiple-choice questions, and they sometimes ran evaluations on a subset of the whole question set. They

had another question set containing direct-answer questions, but in our study, we focused on multiple-choice questions as they were the center of solver development at the time of the study.

To evaluate a solver, the QA developers ran the solver on an arbitrary set of questions. The solver took each question's description and answer choices, and produced a confidence score for each choice. The choice with the highest confidence score was taken as the solver's final answer to the question. Sometimes, confidence scores of choices were tied, and the solver returned more than one choice as its answer to a question. Currently, QA developers measure the solver's overall accuracy in answering these questions. In addition to single solver evaluation, QA developers also ran multiple solvers together as an ensemble solver. The output of an ensemble of solvers combined individual solvers' answers and returned an answer.

3.3.1.2 QA developer goals and tasks

From interviews and informal interactions with the QA developers, we noted three primary goals for them when analyzing evaluations:

G1. Look for ways to improve a solver. When they analyze evaluations, one key goal is to seek opportunities to improve the solver. Examples include examining whether a solver has enough data (e.g., knowledge) to answer a question, or whether dependencies such as entailment service (i.e., an NLP service to extract relations in text) impact solver performance.

G2. Examine if new changes to a solver improve performance. After QA developers modify a solver, they evaluate whether the changes are helpful. As they often have a hypothesis in mind about how the changes may work, they also examine whether the

newer version of a solver behaves as expected, and whether any unexpected behavior appears.

G3. Understand the strengths and weaknesses of solvers. Since the final QA system combines the answers of all solvers, it is important to understand the strengths and weaknesses of the solvers. This understanding provides opportunities for a researcher to better contribute to overall system performance.

We then collected and compiled the following common tasks by observing how QA developers investigate evaluations to achieve the above goals. For each task, we note in parentheses its corresponding goals.

T1. Compare two or more versions of a solver for G1 and G2. After modifying a solver, QA developers usually compare the new version with the original version. They look for questions that the newer version gets right but the older version gets wrong, and vice versa. Sometimes, they investigate questions for which the two versions of the solver have different answers, regardless of the correctness. Since solver development is not a linear process (i.e., versions can branch out), these comparison tasks are sometimes performed on more than two versions at the same time.

T2. Contrast one solver's behavior with that of other solvers (G1, G3). In addition to comparing different versions of a solver, QA developers also contrast a solver's behavior with that of other solvers to understand solver strengths and weaknesses (G3). QA developers compare their solvers' behavior to that of others as a reference to show increases or decreases in their own solvers' performance.

T3. Categorize and/or filter questions (G1, G2, G3). QA developers often focus on question subsets to improve certain categories (G1), investigate component-wise changes (G2), or identify patterns of solver performance between subsets (G3). To this end, they either try to group the whole set into categories and examine individual categories, or they find a way to retrieve relevant questions (e.g., questions containing certain keywords).

T4. Investigate question descriptions to make sense of a solver's behavior (G1). QA developers usually read question text and answer choices first to see if they can come up with a hypothesis for the solver's correct or incorrect answer. This task relies on their past experience in developing the solver and debugging similar questions. QA developers do not always have an idea about why solvers behave in certain ways for a question. Therefore, they must closely investigate the question to discover situations where, for instance, a solver is being misled by a particular word and thus exhibits extremely different behavior for two very similar questions.

T5. Inspect how a solver answers a question (G1, G2). To know more about how a solver answers a question, QA developers program solvers to produce intermediate output (e.g., describing the reasoning process) so that they can quickly make sense of what may have happened. They also have solver-specific tools to drill down into detailed steps.

T6. Find insight that can improve a large subset of questions at once (G1). One QA developer noted, "We want to make changes that can fix (solvers' answers to) many questions, not just one at a time." In order to efficiently gain insight, they typically

investigate subsets of questions that have shared properties and that are not too small (i.e., fewer than 10 questions). For instance, they examine questions that belong to the topic ‘adaptation’ to figure out what knowledge should be added to the knowledge base to answer questions on this topic.

T7. Look for questions for which solvers return no answers, or multiple answers (G1). To improve a solver, one common approach taken by researchers is to investigate questions for which the solver returns no answers, or multiple answers. For these questions they seek ways to modify the solver to break ties between choices, or cause the solver to return a single answer.

T8. Share their findings and insights with other researchers (G1, G3). Researchers sometimes share findings from evaluation analyses with each other. In particular, when discussing solver strengths and weaknesses, spreadsheets and screenshots are popular ways to illustrate their findings.

3.3.2 *Preliminary investigation: Sunburst visualization*

During our user requirement analysis, the QA developers expressed interest in utilizing the three types of categories they manually labeled: topic, subtopic, and question type (To be consistent with later sections, we term each categorization an angle. Note that angles do not refer to degrees encoded within the Sunburst). Thus, as a way to crystallize user requirements, we created an interactive Sunburst visualization [Stasko and Zhang 2000] using the three angles (Figure 3.1). In this visualization, a Sunburst corresponds to an evaluation. The three angles were arranged hierarchically. Each layer of the Sunburst consisted of the categories within one angle. The color indicated the solver accuracy in the corresponding categories in an evaluation. The

accuracy was bucketed into 10 levels of the red-green scale shown in the legends on the top left. All Sunbursts were linked together such that as users interacted with one of the Sunbursts by clicking a category in any level, the other Sunburst views were updated to reflect the change.

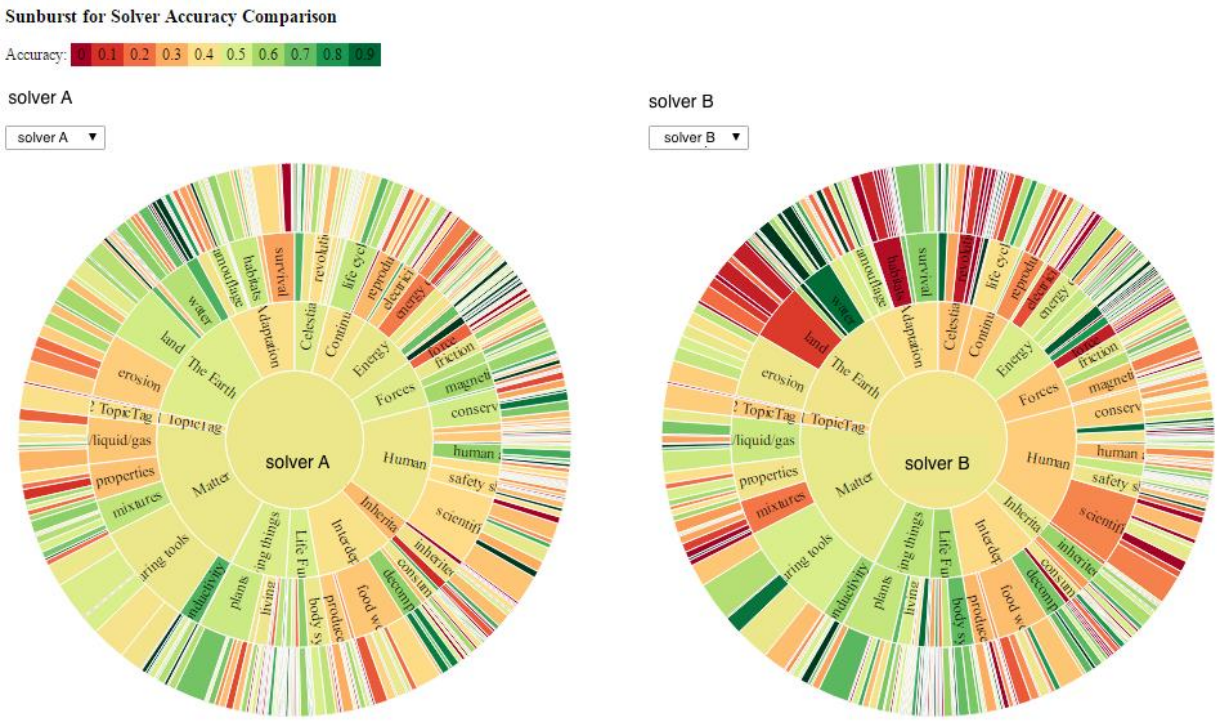

Figure 3.1: Sunburst visualization utilizing three angles: topic, subtopic, and question type, the three question categories. Each Sunburst layer consists of the categories within an angle. Color indicates solver accuracy for the corresponding categories in an evaluation.

3.3.2.1 Challenges

We presented the visualization to the QA developers and conducted contextual inquiries. Although researchers reacted positively and commented that some solver patterns confirmed their understanding of the solver strengths and weaknesses, they indicated a few challenges and issues when using this Sunburst visualization for their analysis:

C1. Difficulty when comparing more than two evaluations. As tasks T1 and T2 indicate, QA developers must compare more than two evaluations. We attempted to present more than two Sunbursts to researchers in a single view: Although the visualization did provide an overview of the results of multiple evaluations, the more Sunbursts that were shown (i.e., the more evaluations loaded), the larger area they occupied on the screen. This made it more difficult to conduct complex analysis and comparisons.

C2. Unidentifiable categories in outer layers. Although we used only a three-level hierarchy in this Sunburst visualization, the researchers still found it difficult to see and identify categories in the outer layers. Although they could filter categories in outer layers of the hierarchy, finding the category they sought to investigate was challenging. This problem was further complicated by additional angles, which added more layers to the Sunbursts.

C3. Complex query requirements. Though the Sunburst visualization provided an overview of the evaluations, QA developers found it difficult to perform more complex queries. For example, they may want to filter on multiple angles, such as questions that belong to the topic ‘Matter’ or ‘The Earth’, but that also belong to the question type ‘Definition’ or ‘Story’. Such complex queries were difficult to perform using the Sunburst visualization.

3.3.3 *Design rationales*

Given the identified goals and common tasks of users, as well as the challenges found in our preliminary investigation of the Sunburst visualization, we present the following design rationales that led to our focus on prismatic analysis and the QSAnglyzer design.

R1. Take questions as the center of analysis. When QA developers analyze evaluations, questions are the center of analysis. In most of their tasks, they directly investigate questions (T3, T4, and T6) or solver behavior with regard to questions (T5 and T7). In addition, questions are the only aspect of the QA system shared between all solver evaluations. Furthermore, even though solvers can be very different from each other, and not every researcher knows all the solvers, they can still communicate and share insights about the questions (T8).

R2. Prioritize accuracy, the number of questions, and accuracy variance as key variables. When researchers modify a solver, their goal is to increase accuracy (G1). When they investigate a subset of questions, they want to find a large subset to ensure that changes made based on the set have a broad impact (G1 and T3). In addition, when studying solver strengths and weaknesses (G3 and T2), they look for a set of questions for which solvers have highly variant accuracy. Therefore, we consider accuracy, the number of questions, and accuracy variance as the high-priority variables for design.

R3. Support categorization. Researchers group questions into categories when focusing on a subset of questions (T3), comparing between versions of a solver (T1), or contrasting solvers (T2). Thus, we observe that categorization can be valuable in revealing patterns in evaluations.

R4. Consider diverse comparison scenarios and complex queries. QA developers need support for diverse comparison scenarios and complex queries. They compare between versions of solvers (T1) and different solvers (T2). They also want to determine whether a solver's performance is worse in one category than in another (T3), to decide

which category should be improved (G1). Moreover, they also compare whether two categories have different performance patterns across solvers (T3) to identify strengths and weaknesses (G3). These comparison scenarios can take place after search and filtering (T3 and R5). Therefore, it is necessary to have a design that supports any combination of the above comparison scenarios.

R5. Enable filtering and search. As researchers often drill down into question subsets (T3), our VA design should include both filtering and search functionalities.

R6. Show multiple angles at the same time. QA developers often seek to examine evaluations from multiple angles. For example, they may seek questions for which two versions of a solver produce different answers (T1). They may also seek to determine for which topics solver accuracy differs the most (T2). Hence, it is critical to show multiple angles of evaluations. In addition, from challenges C2 and C3 we learn that presenting different angles as layers in a hierarchy may not be ideal, and thus we suggest presenting multiple angles in parallel.

R7. Ensure scalability. As there may be more than two sets of evaluations involved in the analysis (C1), it is essential that the design scales up. In our case, we aim to support at least eight sets of evaluations, as the QA system contains eight solvers. Additionally, we must ensure that the design scales to multiple angles (R6).

R8. Make different levels of evaluation details available. The design should incorporate various levels of evaluation details, including high-level metrics such as accuracy, the description and choices of questions (T4), solver confidence scores on each choice of a question (T7), and intermediate outputs (T5).

R9. Accommodate existing workflows. Our visual analytics tool design should fit into the QA developers' existing workflows so that they can use the tool together with their solver-specific tools (T5), and discuss and share findings with each other (T8).

3.4 Prismatic Analysis

The study of QA developers' current evaluation analysis workflows underlies our proposal of *prismatic analysis*. Although this is in the context of QA systems, we argue that the approach can be applied to other types of complex AI system evaluation analysis. Therefore, we start from a generalized formal definition of prismatic analysis using mathematical notation:

Given a set of data points that constitute a high-dimensional *data space* D, we define a set of categorization functions F. Each function $f_i \in F$ divides D into a set of categories C_i from the i-th angle. In addition, we define a set of aggregate metrics M. Each metric $m \in M$ represents aggregated information of data points in scalar. As a result, we can compare categories within an angle or between angles based on the set of aggregate metrics. Furthermore, as some of the metrics may have multiple measurements, we can also compare these measurements within a category. Last, we can filter to a subset of D and conduct the same analysis.

We call this approach 'prismatic analysis' because when we consider the data space D, the categorization functions F act like a set of prisms. These prisms are oriented at different angles toward the space so that we see the space divided in various ways. This results in multiple ways to group the data points into categories. Then the set of aggregate metrics M summarizes all the categories in the same way regardless of angles; this aids us in making comparisons within and between categories in an angle, as well as making connections between angles.

We then describe how prismatic analysis can be applied to analyze QA evaluations: As questions are the center of QA evaluation analysis (R1), the question space, the high-dimensional space formed by questions, corresponds to the data space D. The question space can be divided into categories from multiple angles (each angle corresponding to a categorization function in F), such as by topics or by question types. This implements our design rationales R3 and R6. The aggregate metrics M here include accuracy (denoted as M_{acc}), the number of questions (M_{num}), and accuracy variance (M_{var}) between evaluations, as suggested by design rationale R2. The M_{acc} metric can have multiple measurements when we load multiple evaluations. Moreover, prismatic analysis allows us to filter subsets of questions to conduct the same analysis, which fulfills design rationale R5.

Prismatic analysis is not limited to QA system evaluations. This approach can be applied to other types of complex AI system evaluation analysis by identifying the data space D, categorization functions F, and aggregate metrics M. We further discuss this point in Section 3.7.4.

Although prismatic analysis captures many analytical tasks performed by QA developers, it can be overwhelming to have multiple angles and metrics in pure text (e.g., labels and numbers). In addition, design rationales R7, R8, and R9 are not directly supported by prismatic analysis. This leads to the design of QSAnglyzer, our visual analytics tool introduced in the next section.

3.5 QSAnglyzer

In this section, we describe the design of QSAnglyzer, which is implemented as a JavaScript web application using React.js and D3.js. All data are stored in a PostgreSQL database and can

be accessed through web service calls (implemented in Node.js). QSAnglyzer aims to support prismatic analysis of QA evaluations; its design follows the rationales we derived in Section 3.3. To be more specific, QSAnglyzer enables QA developers to load evaluations into the tool, and supports visual comparison and interactions based on the framework of prismatic analysis. The system treats each evaluation as a dataset in which each question is a data point. Results from different evaluations are different dimensions of the data points. The system provides seven default angles (F) to divide questions into groups; we explain the details of each angle in Section 3.5.1. After the user loads an evaluation on the system or performs filtering or search, the system automatically calculates the aggregate metrics (M_{acc} , M_{num}, and M_{var}) for each category. The metric M_{acc} has N measures, one for each evaluation, where N is the number of evaluations loaded on the system.

3.5.1 *Angles*

By default, the system provides seven angles that can be divided into three types: question-related angles, evaluation-related angles, and free labels. These angles are chosen and defined based on what we found QA developers need, although we envision that more angles will be defined in the future. Question-related angles utilize the three sets of manually labeled categories from the QA developers (topics, subtopics, and question types) used in Section 3.3.2.

Evaluation-related angles are a set of angles automatically derived from evaluations. These include correctness (is correct), agreement of answers (difference), and the number of selected choices (num selected). The ‘is correct’ angle categorizes questions based on whether a selected choice was correct (T) or not (F), or having no answer (X). For multiple evaluations, the categories are correctness permutations across evaluations. For example, when the system is

loaded with three evaluations, the ‘F / T / T’ category refers to questions for which the first evaluation is incorrect but the second and third evaluations are both correct.

The ‘difference’ angle divides questions into two categories: ‘same’ and ’different’. Questions for which all evaluations have the same answer belong to the former; the rest belong to the latter. A question may have the same answers in two evaluations, but as long as there is another evaluation that has a different answer, it belongs to the ‘different’ category.

The ‘num selected’ angle aims to support task T7 by categorizing questions based on the number of selected choices. For simplicity, we treat questions with more than two choices as being in the same group; thus each question can only have ‘0’, ‘1’, or ‘>=2’ selected choices. As with the ‘correctness’ angle, in the case of multiple evaluations, the categories are permutations across evaluations. For instance, when there are three evaluations, the category ‘>=2 / 1 / 1’ includes questions for which the first evaluation selected more than one choice, and the other two selected only one choice.

The ‘free labels’ angle is designed for users to assign customized categories to questions. Users label the questions in the bottom ‘Question Table’ panel. We describe this process in Section 3.5.2.3.

3.5.2 *Interface design*

Figure 3.2: The QSAnglyzer interface consists of three basic panels: 'Evaluations' (top left), 'Question Space Angles' (top right), and 'Question Table' (bottom) panels, following the well-known visualization mantra: "Overview first, zoom and filter, details on demand"

As shown in Figure 3.2, the QSAnglyzer interface consists of three basic panels: the top left 'Evaluations', the top right 'Question Space Angles', and bottom 'Question Table' panels. This design follows the visualization mantra: "*Overview first, zoom and filter, details on demand* [Shneiderman 2003]." The 'Evaluations' panel provides the most high-level overview of the evaluations, and helps users keep track of how many questions have been filtered by showing the partial bars. In the 'Question Space Angles' panel, users perform prismatic analysis; this panel allows users to filter and zoom into a finer set of questions. Finally, the 'Questions Table' panel shows details related to questions and solver behavior, and supports keyword search and question bookmarking (i.e., starring). This design fulfills our design rationale R8. In addition to the basic panels, users obtain a link to share the current view of the tool by clicking the link icon on the

top right corner of the tool, which follows design rationale R9. Below, we explain each component of the QSAnglyzer and the design decisions for visual encoding and tool interaction.

3.5.2.1 Top left panel: a visual summary of evaluations

In the top left ‘Evaluations’ panel, users add or remove evaluations to visualize by providing URLs (the URLs link to their existing database APIs). Users assign a name to the evaluation, or the system automatically names the evaluation based on the solver. We represent each set as a colored vertical bar, where the color indicates the overall accuracy (dark green represents high accuracy). The maximum height of the bars corresponds to the highest number of questions loaded to the tool, and the height of the individual bars are proportional to the number of questions in each evaluation (labeled in text under the bars). When hovering over the bar, a tooltip pops up to show the exact accuracy.

These bars provide a visual summary for evaluations, and are updated when the user interacts with the tool. When a user filters a set of questions, a partial bar appears reflecting this filtering action. The color of the partial bar represents the accuracy for the filtered set of questions. If the number of questions varies across evaluations, the system automatically filters the subset of questions that all evaluations contain.

Design choices. Although color is not the best choice for encoding quantitative values such as accuracy, it is a visual attribute supported by human pre-attentive processing [Ware 2012]. In addition, since length is a recommended visual attribute for showing quantitative data [Munzner 2014], we use it to encode the number of questions. The two design choices are also used in the ‘Question Space Angles’ panel.

3.5.2.2 Top right panel: Question space angles

The Question Space Angles panel on the top right supports prismatic analysis as a metaphorical representation of the question space. Each box on the panel represents an angle (a $f_i \in F$) and its categories (C_i), and each category is represented by a horizontal slice. The height of a slice corresponds to the number of questions (M_{num}) in this category, where the height of an entire angle box corresponds to the number of visible questions (i.e., all questions or remaining questions after filtering). For simplicity, hereafter we directly use 'angles' to refer to these boxes, and 'categories' to refer to the horizontal slices.

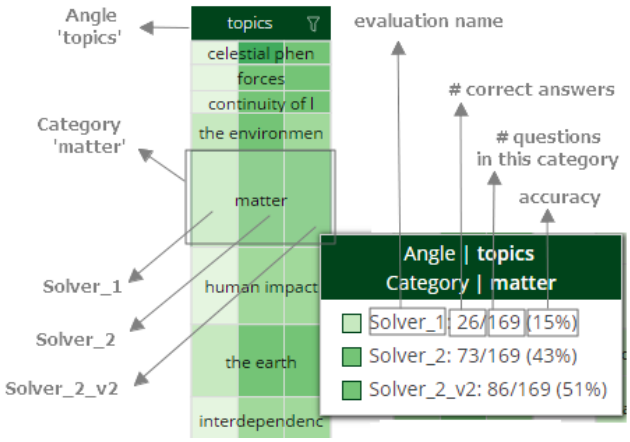

Figure 3.3: The design of an angle. The header shows the name of the angle. Each slice is a category within the angle, and its height corresponds to the number of questions in the category. The vertical segments represent evaluations, and the colors indicate how accurate the solver in the evaluation is on the questions in the category. When hovering over a category, a tooltip appears, providing detailed numbers.

Figure 3.3 illustrates the parts of an angle. When there are *N* evaluations, every category is divided into *N* vertical segments. Each segment represents one evaluation. For example, in Figure 3.2, as the tool is loaded with three evaluations, each category has three segments. The

order of the segments follows the order in the ‘Evaluations’ panel. The color of each segment reflects the accuracy (M_{acc}) of the solver in an evaluation on the questions within the category.

When the user clicks a category within an angle, a filtering action is triggered, and all angles are updated based on the remaining questions. When users hover over a category, a tooltip appears (Figure 3.3), providing detailed information such as the number of questions in the category, the number of correctly-answered questions in each evaluation, and the corresponding accuracy. When users seek to filter more than one category, they use the filter box shown in Figure 3.4A. If a filter is applied to an angle, the filter icon on its header turns white. The ‘Active Filters’ panel on the left lists the currently applied filters.

Categories within an angle are sorted with respect to variances in accuracy (M_{var}) across evaluations (i.e., the first category on the top has the highest variance across evaluations). In the filter box, however, we sort the categories with respect to the number of questions (M_{num}). As we have observed cases in which both the order of M_{num} and M_{var} can be useful, we use different ordering mechanisms as a quick way to provide the two types of information without re-ordering.

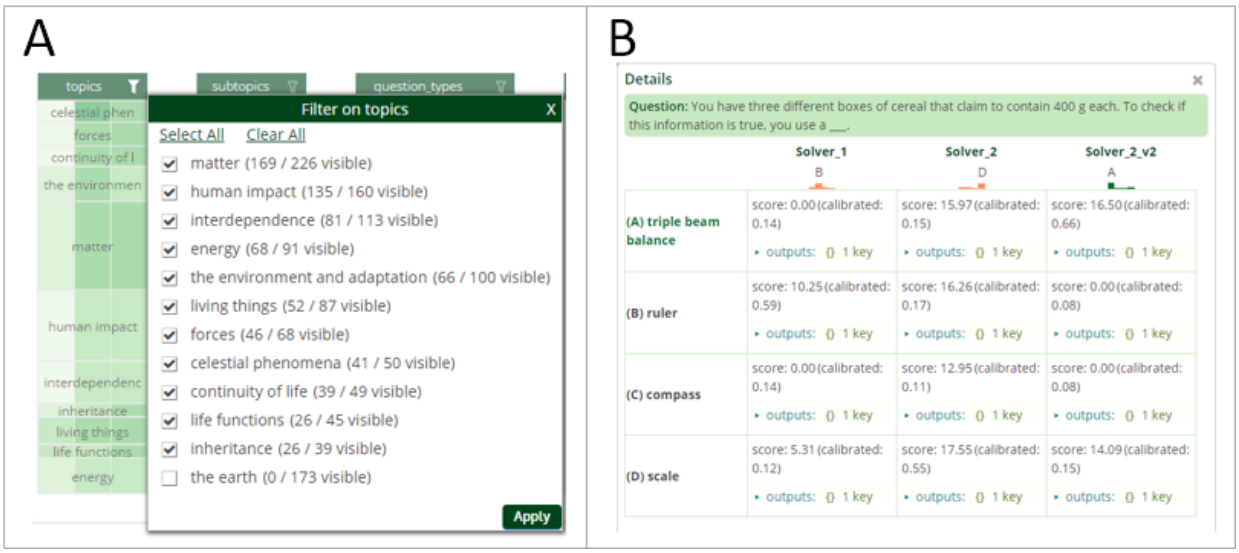

Figure 3.4: (A) The filter box of the 'topics' angle. The user can filter on more than one category using the check boxes on the left. The categories are sorted with respect to the number of questions. (B) The panel shows the details of the solvers' answers to a question. By unfolding the 'outputs', users examine a solver's intermediate outputs for a choice.

Design choices. We chose to use a box to represent an angle and show all angles in parallel to reflect challenge C2 and design rationales R6 and R7. Such a design enables the easy adding and removing of angles, as well as changing their order. In addition, putting evaluations as segments next to each other greatly facilitates comparisons between evaluations and reflects challenge C1. We easily scale up to nine evaluations (R7). In addition, we chose to use orders to encode M_{var}. Although M_{var} is a quantitative variable, the exact values and the difference between values are not critical for analysis. Therefore, we assign an ordinal encoding to the variable. We also use orders to show M_{num} . This is an additional encoding to the height encoding for this variable, which is suitable for ordinal encoding.

3.5.2.3 Bottom panel: Question table

The ‘Question Table’ panel at the bottom provides finer-grained details of solver evaluations. By default, this panel is collapsed. Every row in the table corresponds to a question and solvers’ answers in the evaluations. The first column enables users to ‘bookmark’ (or ‘star’) questions, which can then be viewed as the bookmarked questions in the ‘Starred’ tab. The second and following columns of the table show the solvers’ choices and confidence scores in each evaluation. The order of the evaluations is consistent with both the order in the ‘Evaluations’ panel as well as that in the categories.

Bar charts in these columns represent the normalized confidence scores (between 0 and 1) returned from each evaluation, where the x-axis is the choices and the y-axis is the scores. When users mouse over the bars, a small tooltip appears with the normalized and raw scores. The letter above each bar chart represents the choice with the highest confidence score (that is, the selected choice). When a solver returns more than one choice, the letters of all the selected choices are shown above the bar chart, whereas when a solver returns no answer, the grid is empty (as in the first column in the second row in Figure 3.2). In contrast to the accuracy colors in the top two panels, the colors in these bar charts are binary: green for correctly answered questions, and orange for incorrectly answered questions.

In addition to solvers’ answers, each row of the question table also shows the question’s membership to each angle’s categories (the yellow labels in the bottom panel of Figure 3.2). Users change the question’s membership of a category by adding or removing labels. This is also where users create and assign customized labels to the ‘free labels’ angle.

Currently, each page of the question table contains 100 questions. Users can also perform keyword searches. As with filtering, completed search actions update the interface to reflect the questions returned from the search. If users want to retrieve solvers' intermediate outputs for a question, they click the eye icon in the last column. A pop-up panel (Figure 3.4B) appears so that users can compare intermediate outputs (folded under 'outputs' by default).

Design choices. The design of this panel is based on the spreadsheets the QA developers created before the study. Therefore, even though both colors and heights encode different variables from other panels, for the QA developers these encodings are acceptable. Note that we designed this panel for multiple-choice questions, but the design can be tweaked to show direct-answer questions without the need to change the other panels.

3.6 Evaluation

To verify the effectiveness and efficiency of QSAnglyzer, we conducted a lab study with both non-expert and expert reviewers. The goal of the lab study with non-experts was to evaluate the effectiveness of the visual and interaction design of QSAnglyzer. We defined the tasks in the lab study based on tasks drawn from the requirement analysis. With the expert reviews, we sought to examine whether QSAnglyzer facilitates insight discovery. The two evaluation approaches were designed based on Munzner's nested model [15]. In this section, we describe both studies in detail.

Table 1: Lab study design. We took the number of evaluations ($IV_{NumOfEval}$) and the tool used (IV_{Tool}) as the independent variables.

		Tools (IV_{Tool})	
		Excel (control)	**QSAnglyzer (treatment)**
# Evaluations ($IV_{NumOfEval}$)	2	2-Excel	2-QSAnglyzer
	4	4-Excel	4-QSAnglyzer
	6	6-Excel	6-QSAnglyzer

3.6.1 *Lab study with non-experts*

Since QA developers commonly utilize spreadsheets in their workflows, we designed a set of Excel spreadsheets that support the same functionalities for prismatic analysis. These functionalities include the use of multiple angles to categorize questions; the use of accuracy, the number of questions, and accuracy variance as aggregate metrics; and filtering. In addition to the interface, we tested whether the number of evaluations loaded in the tools impacts user performance.

3.6.1.1 Lab study design

Participants were asked to perform sets of predefined tasks with both QSAnglyzer and the Excel spreadsheets. In addition, we tested both Excel spreadsheets and our tool with 2-, 4-, and 6-solver evaluations (details in Table 1). We took as the independent variables the number of evaluations ($IV_{NumOfEval}$) and the tool used (IV_{Tool}). For each condition, the participants performed three tasks; we measured the time spent (DV_{time}) and the accuracy (DV_{acc}) of these tasks. We adapted a within-group study design in which all participants went through all

conditions. We used balanced Latin square design [Williams 1949] to minimize the learning effect.

3.6.1.2 Data and tasks

We prepared three sets of evaluations from different solvers or versions of solvers to design tasks. The tasks were selected from the common workflow of QA developers. Here is an example task and what it was meant to achieve for QA developers:

"For questions for which the three solvers yield different answers, filter to the category "F/T/F" of the "is correct" angle. What is the category of "question type" that has the largest number of questions?" (This task was a combination of tasks T1, T3, and T6.)

The goal of this task was a combination of goals G2 and G3. QA developers are interested in questions that solvers disagree on, and specifically look at those question types for which a solver outperforms others.

3.6.1.3 Excel spreadsheets

We created highly customized Excel spreadsheets that support prismatic analysis (Figure 3.5). For example, as shown in Figure 3.6, aggregate metrics of each categories were also pre-calculated and presented. A data table on the first sheet enabled users to filter on angles. Upon filtering, aggregate metrics were automatically updated. However, unlike QSAnglyzer, the order of the categories in the spreadsheet was not re-ordered after filtering. We believe this is a reasonable difference and limitation, since automatic re-ordering requires the writing of Excel-specific scripts, which for most users is not an option.

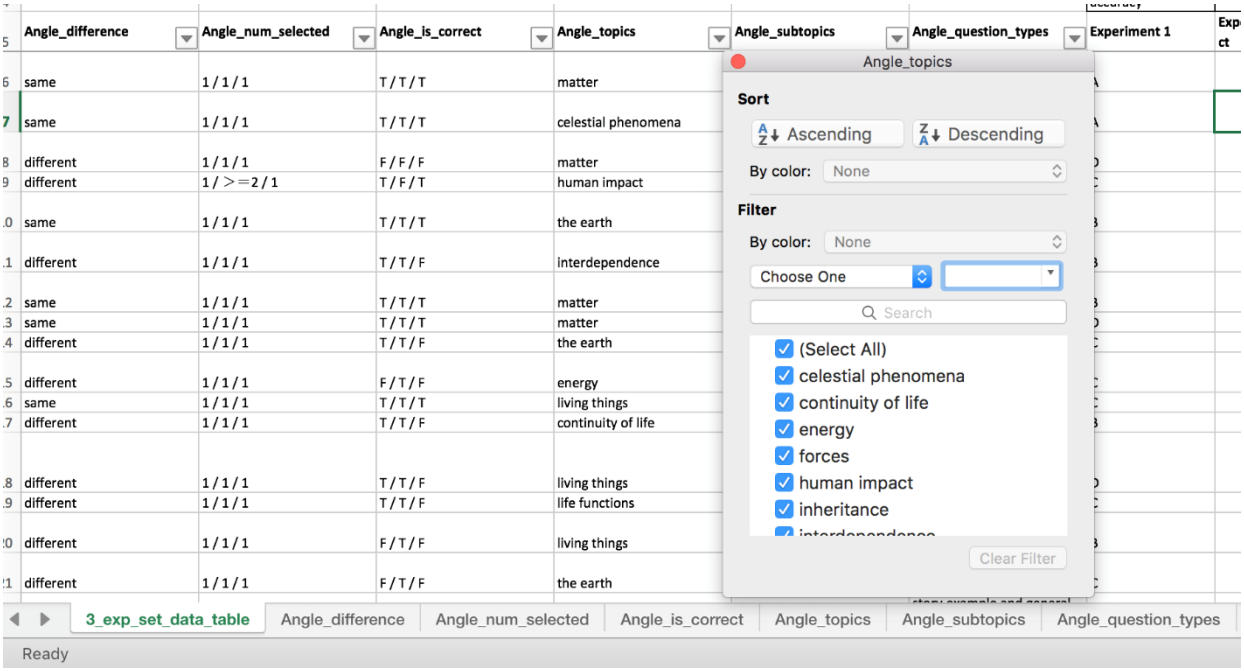

Figure 3.5: Screenshot of highly customized Excel spreadsheet for baseline condition. Users can perform rich interactions: for example, users can filter to a category or categories, and the summarized performance statistics of each category are pre-calculated and presented.

Angle_question_types	Category Statistics		Evaluation 1		Evaluation 2		Evaluation 3	
	accuracy variance across evaluations	# visible questions for categories	# correct	accuracy	# correct	accuracy	# correct	accuracy
other	0.006903353	26	12	46%	13	50%	8	31%
definition	0.006130787	163	117	72%	98	60%	86	53%
single fact	0.004566576	316	204	65%	210	66%	162	51%
conjunction of facts	0.003218823	147	81	55%	78	53%	62	42%
example and general concept	0.003212867	188	93	49%	99	53%	74	39%
causality	0.002771219	308	171	56%	158	51%	132	43%
comparison	0.002418306	191	85	45%	96	50%	73	38%
story	0.001628439	188	91	48%	89	47%	74	39%
	Total # visible	1527						

Figure 3.6: Aggregate metrics in customized Excel spreadsheet for categories of 'topics' angle. Upon filtering, aggregate metrics are automatically updated.

3.6.1.4 Participants

From the research institute and from our social networks, we recruited 13 non-experts who were not QA developers but had experience in AI-related fields. The ages of the participants

ranged from 21 to 40 (9 male, 3 female, and 1 other gender). Ten held post-graduate degrees in computer science or related fields.

3.6.1.5 Study setup and process

Each participant came to the study room individually. We described the context of the project to them, and then explained the two tools: QSAnglyzer and the customized Excel spreadsheets. Prior to audio and screen recording, the moderator secured participant permission. All tasks were performed on a 28-inch external monitor with a USB mouse and keyboard.

Prior to the formal sessions, the participants were given four practice tasks: two on each tool. We guided the participants in the use of both the Excel spreadsheets and QSAnglyzer, and ensured they understood the tasks. The formal study consisted of six sessions corresponding to each study condition. In each session, the participants were given one of the two tools preloaded with N evaluations. The number N depended on which condition the session was. During each session, the participants performed three tasks which were randomly selected from a pool of predefined tasks without replacement. Each task had a three-minute time limit. We ensured that participants spent no more than the allotted time on the tasks.

During the practice session, the participants spoke aloud their task answers directly. Once the formal sessions started, the participants were required to submit their answers via a Google form in a window next to the tool. After performing each task, we helped participants restore the tool to its initial state with no filtering and sorting.

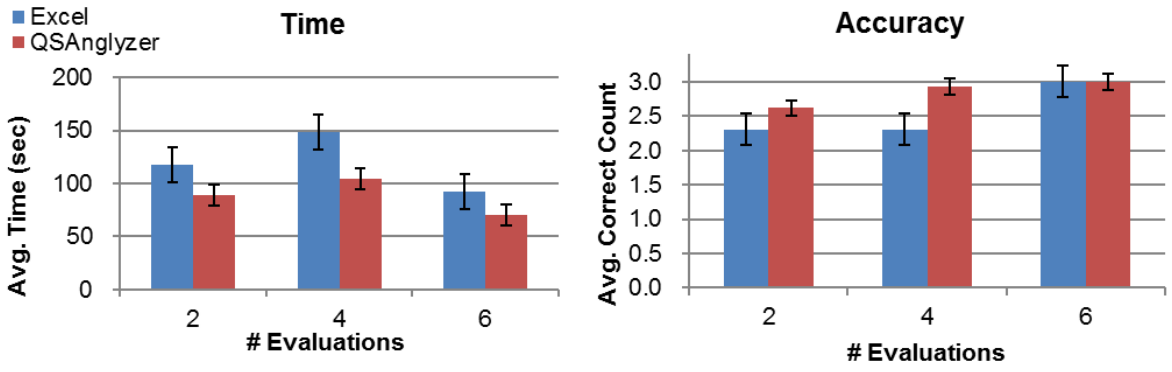

Figure 3.7: Average time and accuracy for test conditions. Statistical testing showed significant differences between tools (IV_{Tool}) on both DV_{time} (p < 0.01) and DV_{acc} (p < 0.05).

3.6.1.6 Analysis and results

We analyzed the results of the lab study by a mixed-effects model analysis of variance. The model took the within-group variables (IV_{Tool} and $IV_{NumOfEval}$) as factorial effects, but considered each participant as a random level drawn from a population [Wolfinger et al. 1991]. For each condition, the time (DV_{time}) of each participant was the average time for the completion of three tasks, whereas the accuracy (DV_{acc}) was the number of correctly answered tasks (up to three). The average time and accuracy for each condition are shown in Figure 3.7. Statistical testing showed significant differences between the tools (IV_{Tool}) for both DV_{time} ($p < 0.01$) and DV_{acc} ($p < 0.05$), but the difference between the number of loaded evaluations ($IV_{NumOfEval}$) was significant with respect to neither time nor accuracy. There was also no interaction effect between IV_{Tool} and $IV_{NumOfEval}$. These results showed that QSAnglyzer more effectively and efficiently supported users performing the tasks. Interestingly, the results did not show significant differences between three levels of $IV_{NumOfEval}$. This may be because the tasks did not emphasize the comparison of multiple evaluations. Even though the results were not

significant, during the study we did observe that when more evaluations were given, participants expressed difficulty in finding the target categories.

3.6.2 *Use cases from expert reviews*

We invited five QA developers in the research institute to test and review QSAnglyzer. Each spent an hour testing the tool, and we asked them to think aloud throughout the testing sessions. We preloaded the evaluations of the solvers they developed, and let them explore freely. The researchers all reacted positively and found numerous insights within an hour. They commented that being able to thoroughly examine questions from various angles enabled them to quickly form hypotheses and verify them. They also starred many questions that they were interested in for further inspection. Below we highlight some of use cases and insights they found during the sessions.

3.6.2.1 Investigating gain and loss of accuracy after changes

As noted in goal G2, QA developers seek to examine how changes to a solver impact its behavior. The researchers performed this task using QSAnglyzer. For example, one researcher first filtered to the ‘different’ category in the ‘difference’ angle, and then investigated the ‘is correct’ angle (Figure 3.8A) by comparing the height of the categories. The researcher then explained: *“This (the “is correct” angle) shows that we have gained more than we lost after adding the new knowledge base... which is a good thing. It also shows that many questions our solver could not answer before (marked as “X” in the categories’ names) now have answers.”*

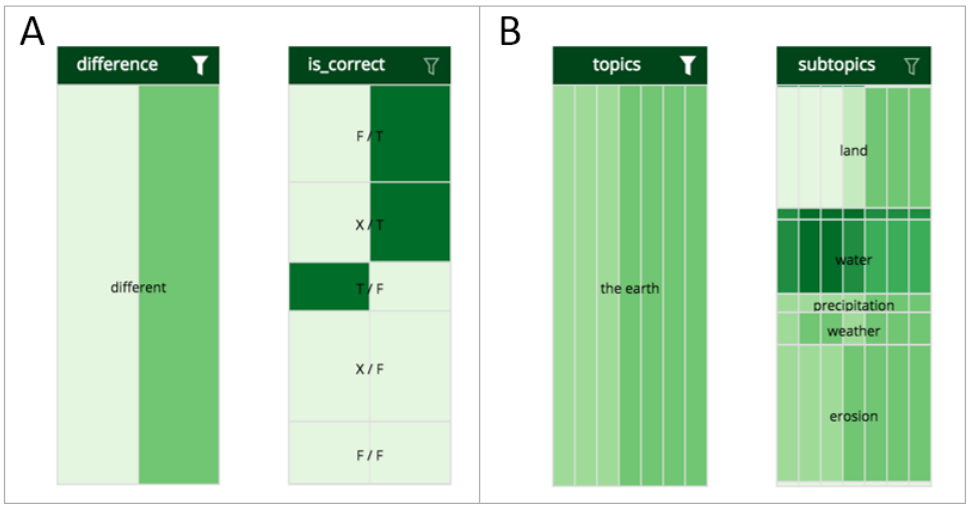

Figure 3.8: User insights: (A) One of the QA researchers confirmed their solver was improving in the newer version (the right segments) by examining the 'difference' and 'is correct' angles. After filtering to the 'different' category, the researcher was able to draw this conclusion by comparing the number of questions in the 'X / T' and 'F / T' categories with the number of questions in the 'T / F'. (B) Another QA researcher discovered an unusual trend over time in the 'water' category under the 'subtopic' angle. The tool was loaded with seven different versions of a solver (in chronological order from left to right). The researcher filtered to the 'the earth' category in the 'topic' angle, and found this trend in the 'subtopic' angle. After further examination of the intermediate outputs, the QA researcher concluded that the earlier versions had higher accuracy in this category only by chance.

3.6.2.2 Uncovering incidentally correct questions

In one session, the tool was loaded with the evaluations of seven versions of a specific solver, from left to right in chronological order. Figure 3.8B shows a pattern discovered by a researcher in this session; the researcher learned that the solver's overall performance gradually improved for questions on the topic "the earth" due to new knowledge added to the solver's knowledge base. However, the researcher noticed that the subtopic "water" did not show the same improvement. By filtering for questions with the "water" category, the QA developer

concluded that the first couple of versions of the solver were answering this set of questions correctly only by chance. This insight was novel and thought-provoking to the QA developer.

3.6.2.3 Discovering previously unknown solver behavior

During the evaluation sessions, three QA developers observed solver behavior that surprised them. For instance, because of the "num selected" angle a researcher realized that an earlier version of his solver chose more than one answer choice for many questions (more than 30). He noted that this pattern is significant since the tie between answer choices had been broken by newer versions of the solver. Another researcher commented that he did not know there were questions where his solver did not choose any answer choice at all. He also noted that he discovered a potential bug using QSAnglyzer: he believed that the sum of all confidence scores of all answers should equal one, which was not always the case.

3.6.2.4 Relating to different solvers

In addition to loading different versions of the same solver, we also showed evaluations of different solvers to the QA developers. One found that a legacy solver that used an IR-based approach tended to perform better on "story"-type questions (a category of the "question types" angle). Another QA developer compared two versions of a solver along with a legacy IR-based solver as a reference. By doing so, the QA developer learned that the solver behaved much like the legacy IR-based solver: "*I am interested in questions that used to be wrong, and now are correct, and want to see if the IR-based solver also answered correctly on those questions. (after the exploration and comparison)... I guess my solver is now becoming more like an IR-based solver.*"

3.6.2.5 Finding tricky or mislabeled questions

Since QSAnglyzer can be used to quickly form complex queries on different angles, all researchers found questions that they considered tricky or bad (e.g., they thought there was more than one correct answer). One found a few mislabeled questions, and was able to use the label modification functions in the ‘Question Table’ panel after the tool was deployed to their work environment.

3.7 Discussion and Future Work

In this section, we discuss how prismatic analysis and QSAnglyzer scaffold evaluation analysis. Then we discuss potential directions to extend the tool and other future directions of research.

3.7.1 *Scaffolding evaluation analysis workflows*

Since QA or other AI systems are often highly complex, generating insights from evaluations requires scaffolding diverse scenarios for comparison and examination. Prismatic analysis provides a foundation that supports such scenarios by multi-angle categorization. The shared aggregate metrics across angles enable diverse scenarios of comparing categories. The QSAnglyzer design further supports the analysis workflows by leveraging the power of visual comparison and interactive filtering.

From the lab study we found that when performing the comparison tasks using QSAnglyzer, users are faster and more accurate. From expert reviews we witnessed QA developers gain insight into their solvers within one-hour sessions. In addition, when having multiple angles in separate spreadsheets as in our lab study, we found users often lost track and forgot what filters

had been applied. In contrast, QSAnglyzer helped QA developers to maintain context by presenting angles and filters at the same view. This aligns with design rationale R6.

3.7.2 *Adding evaluation-based or automatic angles*

The current QSAnglyzer design includes only seven angles. Though these angles are defined based on the QA developers' need, there is space to add more angles. Three directions for future work include: evaluation-dependent, semi-automatic, and fully-automatic angles. Evaluation-dependent angles refer to angles that are specific to an evaluation, unlike the current evaluation-related angles, which treat all evaluations equally. For example, solvers usually have features for intermediate processing: we can group questions based on these features. However, features are evaluation-dependent and may vary widely across evaluations, which requires further studies on categorization and interface design. Semi-automatic angles are an augmented version of the 'free labels' angle. Although QSAnglyzer supports manually-created angles, researchers commented that they want the system to recommend similar questions when they assign a category label to a question: fully-automatic angles should create categories using clustering algorithms. These three directions are not mutually exclusive. Thus, we can also create evaluation-dependent angles using clustering algorithms, or we can run topic modeling algorithms (e.g., [Blei et al. 2003]) to create automatically generated topics. Preliminary investigation in this direction suggests a challenge: questions that are interesting to researchers may drift from time to time. It may be difficult to collect a set of questions to define other categories that researchers seek to investigate in every evaluation. In addition, automatic algorithms may often result in uninterpretable categories, and thus may not be useful to users. We leave addressing these issues for future work.

3.7.3 *Supporting complex AI system development*

Evaluation analysis is critical in developing complex AI systems, yet it is only part of the development workflow. Many other parts of the workflows are under-studied and could be better supported. For instance, since each solver has many components, we could also build visual analytics to help researchers to understand the interaction between QA system components. In addition, one QA developer told us that QSAnglyzer is useful when they have developed a solver that is accurate enough. When a solver is in its initial development stage, they may want to focus on getting one or two questions correct, rather than on reviewing overall patterns. These examples indicate the need for more holistic studies on complex AI system development, which could lead to an open design space of visual analytics tools.

3.7.4 *Applying prismatic analysis to other AI domains*

One key reason for using prismatic analysis is that the analysis target (e.g., evaluations) and the goal (e.g., improving a QA system) are complex. This necessitates examination from many angles and categorization to help discover patterns. Therefore, prismatic analysis is suitable not only for analyzing QA system evaluations, but also any complex analysis target that can benefit from multi-angle categorization. For example, a potential use case for prismatic analysis is analyzing evaluations of a recommendation system. We can categorize user profiles from multiple angles, and determine how well a recommendation system performs on various profile categories. However, since in AI systems new development practices keep emerging, more in-depth studies are needed on user workflows to apply prismatic analysis in different contexts. In addition, in the QSAnglyzer design, we focus on three aggregate metrics, one of which (M_{acc})

has multiple measurements. Other contexts may have different aggregate metrics, and thus the design of QSAnglyzer may be adjusted. Using prismatic analysis in different contexts and building visual analytics for different contexts are directions that call for further exploration.

3.8 Conclusion

Building AI systems is a fascinating but challenging problem to AI developers, yet most work in HCI and VIS/VA focuses only on the development and investigation of individual models. In this paper, we address the development of complex AI systems in the domain of QA systems. Specifically, we focus on the workflows of multiple-evaluation analysis. We worked closely with QA developers to extract their goals and tasks in evaluation analysis, identifying challenges and design rationales and finally leading to our proposal of prismatic analysis and QSAnglyzer. Although here this is applied to the QA domain, we envision that prismatic analysis can be applied to other AI domains and so on as the design of QSAnglyzer. This work opens a door and highlights the need for better support for AI development. More studies should further examine the design space and seek opportunities for visual analytics research.

3.9 Acknowledgments

We wish to thank all the participants for their time and the anonymous reviewers for their helpful comments. This work was supported by the Allen Institute for Artificial Intelligence (AI2).

Chapter 4. AnchorViz: Interactive Visualization for Error Discovery in Interactive Machine Learning

4.1 Introduction

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

One common approach in iML for selecting unlabeled items is sampling based on uncertainty (e.g., items with prediction scores near a decision boundary) or on a uniform distribution (i.e., items uniformly sampled based on the prediction score distribution). Although these sampling methods can help pull out errors, often the selected items are ambiguous or random errors. Errors with high prediction confidence are not likely to be selected by these sampling algorithms. Hence, model developers may not even be aware of such defects in their models. Furthermore, these sampling methods do not fully leverage the value of human-in-the-

loop. They treat humans as “oracles.” The original meaning of the term “oracle” refers to people who speak for God, not for themselves. In this context, it means humans are only ground truth label providers and do not contribute their own thoughts. Treating people as oracles can cause humans to feel annoyed, lose track of teaching progress, or lose interest [Amershi et al. 2014]. As the discovered errors are mostly uncertain or random, it can be challenging for humans to make sense of the errors to provide useful inputs.

Past discussions have pointed out the benefits of semantic models—models that have semantically meaningful feature representations [Jandot et al. 2016]. Semantic models are useful in helping humans make sense of the model behaviors and ensure their predictions are not based on problematic associations. Specifically in classification problems, humans usually possess knowledge about the target class; they can come up with hypotheses on which underlying “concepts” (i.e., abstract notions that altogether represent the target class) are challenging to the model, and try to fix the errors by providing more labeled items or adding features. However, it is costly and unrealistic to ask humans to provide an exhaustive list of concepts and items, especially for rare cases that are hard to recall. Therefore, careful design is required to efficiently utilize human efforts and maximize the value of each interaction.

In this work, we extend the idea of building semantic models to include the semantic exploration of unlabeled datasets for error discovery and propose *AnchorViz*, a polar-coordinate-based interactive visualization that facilitates this semantic exploration process. Through the visualization, users can create example-based anchors to spread items based on pair-wise similarity. Since users can understand the meaning of the chosen items, they can associate and

contrast items' relations to the anchors. Hence, they can tease apart concepts and find items whose predictions do not align with the corresponding concepts.

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

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

The original paper of this chapter is my internship work in Microsoft Research Machine Teaching team with my mentor Jina Suh, as well as other researchers: Johan Verwey, Gonzalo Ramos, Steven Drucker, Patrice Simard. I proposed the design of AnchorViz. Jina and I implemented the front-end interface, and Johan and Jina implemented the backend server and helped optimize my original hierarchical clustering algorithm implementation. Jina and I conducted the user testing together, and the rest of the authors worked on paper writing and providing feedback throughout the project period. The paper was published in 2018 ACM International Conference on Intelligent User Interfaces (IUI'18) [Chen et al. 2018].

In this chapter, we address our thesis statement by showing that VA tools are helpful not only for traditional AI/ML development paradigms but also for emerging paradigms such as iML. Through the design of AnchorViz and the user study, we demonstrate that VA is a powerful approach to solving challenging issues such as finding feature blindness errors in unlabeled datasets, which is not well supported by computational methods.

4.1.1 *Motivating scenario*

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

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

To illustrate how one can create and use anchors in a classification task, we provide the following scenario: A food blog writer wants to build a binary classifier for finding recipe

webpages. The writer already considers *restaurant menu* pages as negative, so the writer uses existing restaurant menu pages to create a restaurant menu anchor to see other items similar to the concept of restaurant menus. Among the items similar to restaurant menu concept, the writer notices that some items are being predicted as positive. Upon inspection of those predicted positive items, the writer discovers catering webpages. The writer considers catering webpages as not belonging to recipes, labels the discovered *catering* webpages as negative, and creates a catering anchor. With both *restaurant* menu and *catering* anchors, the writer can see if there are other similar items being predicted incorrectly.

4.2 Background and Related Work

In a supervised iML setting, humans build models by providing training items and features in a quick, iterative, and continuous loop [Fails and Olsen Jr 2003]. This model building approach has shown its success in building a decent model using fewer features [Ware et al. 2001]. In addition, it provides meaningful interactions to improve the user's trust and understanding of the system [Stumpf et al. 2009]. This interaction between humans and machines goes beyond simply treating humans as label oracles and requires thoughtful design and careful user studies [Amershi et al. 2014]. However, like traditional supervised ML, in supervised iML, for better generalization performance in the real world, we must also find items to which the current model is blind, which is the focus of this paper.

4.2.1 *Unknown unknowns and prediction errors*

Attenberg et al. [Attenberg et al. 2015] and Lakkaraju et al. [Lakkaraju et al. 2017] use a model's confidence score to categorize prediction errors into known unknowns and unknown

unknowns. Known unknowns are errors for which the model has low confidence; correcting such errors is useful for fine-tuning the system. Unknown unknowns are errors for which the model has high confidence; such errors can be potentially disastrous depending on the problem domain.

Another approach is to characterize prediction errors as feature blindness, ignorance, mislabeling, or learner errors [Meek 2016]. Feature blindness errors arise because the model does not have the appropriate feature representation to learn from the training items. In contrast, ignorance errors are correctly classified if they are added to the training set as the model already has the appropriate feature representation. Mislabeling errors derive from mislabeled items, whereas learner errors refer to issues caused by the configurations of the learning algorithms. In our work, we focus on finding feature blindness errors since these errors represent missing concepts in the current model, which is a type of unknown unknowns.

4.2.2 *Searching for items to label*

There are two types of strategies for searching for items to label: Machine-initiated and human-initiated approaches. The machine-initiated or algorithmic approach uses learning algorithms to suggest items for humans to label so that the model needs fewer training items to perform better [Settles 2012]. Uncertainty sampling is one of the most commonly used active learning strategies. In binary classification, this strategy samples items whose confidence scores are near the decision boundary [Lewis and Catlett 1994, Lewis and Gale 1994]. Active learning strategies are usually not suitable for finding unknown unknowns or feature blindness errors because they often rely on the model's current training results, which cannot overcome the model's blind spots [Attenberg et al. 2015, Lakkaraju et al. 2017, Meek 2016].

Lakkaraju et al. propose an algorithmic approach to find unknown unknowns directly [Lakkaraju et al. 2017]. Their approach leverages systematic biases that are concentrated on specific places in the feature space. Their explore-exploit strategy partitions a dataset based on similarities in feature space and confidence scores, and searches for partitions with a high concentration of items with high confidence. The underlying assumption for their system is that any unknown unknowns introduced to the algorithm can be characterized by the automatically extracted features (e.g., bag-of-words (BoW)), but according to a previous study, using such features for training can have undesired consequences of losing interpretability [Jandot et al. 2016]. In our work, we aim to preserve semantic meanings during exploration to maximize interpretability.

The human-initiated approach allows the humans to find the items that the model should learn. Attenberg et al. introduce the notion of *guided learning* [Attenberg and Provost 2010] and implement *Beat the Machine* (BTM), where crowd workers are tasked to find items from the open world [Attenberg et al. 2015]. Their results show that BTM finds more unknown unknowns compared to a stratified random sampler. Guided learning puts a significant load on users to find adversarial items since recalling is difficult, but when combined with other active learning strategies, this method can help the model learn quickly, especially on skewed datasets. One unique characteristic of BTM is that it leverages the open world as a search space, rather than a closed sample set used in traditional settings. However, this approach does not keep the concepts or structures created and leveraged by users during the search process, so it has limited support for subsequent searches.

In AnchorViz, humans do not play a passive role as labelers of items presented by algorithmic techniques, but take advantage of algorithmic techniques. For example, AnchorViz uses hierarchical clustering (HC) [Sokal 1958] to reduce the user's search space, although ultimately, it is still the user who labels the item. The system also computes the similarity between sets of items in the BoW feature space and presents the results on which users take actions. In addition, we allow the users to externalize their concepts or structures so that they can decompose the target class, reuse previously defined concepts, or redefine and evolve their search strategies as they explore the dataset.

4.2.3 *Visualization for ML*

Visualization is one key approach to facilitating model development. Prior work that supports specific ML tasks usually contains a form of interactive visualization. For example, Talbot et al. design *EnsembleMatrix* for creating and tuning ensemble classifiers [Talbot et al. 2009]. Lee et al. visualize topic models and allow users to modify the models through their interface [Lee et al. 2012]. Ren et al. use a parallel coordinate style of visualization to help examine classification errors [Ren et al. 2016]. However, existing work mostly focuses on discovering errors in the labeled set with little attention to finding errors in unlabeled items.

In AnchorViz, we focus on locating errors in unlabeled items by using visualization to spread out items in a semantic manner. Our design is inspired by *VIBE* [Olsen et al. 1992] and *Dust and Magnet* (D&M) [Soo Yi et al. 2005]. D&M uses the magnet metaphor to attract similar items using pre-defined dimensions of the data. However, the existing dimensions in our iML scenario (i.e., features) may not have any connections with unlabeled items to attract them.

Similar to Adaptive VIBE [Ahn and Brusilovsky 2009], we allow the users to create and refine "anchors" as our version of magnets. We explain the anchors in further detail in the next section.

4.2.4 *Semantic memory and concept decomposition*

In psychology, semantic memory is a type of human memory that stores general knowledge, such as facts and concepts [Reisberg 2013, Tulving 1972]. Network models are commonly used to describe semantic memory; the theory of hierarchical network models [Collins and Quillian 1969] points out that a concept can be decomposed into smaller concepts to store in memory. In ML, concept decomposition is a learning method that leverages clustering [Dhillon and Modha 2003, Dhillon and Modha 2001, Dobsa and Basic 2003]; the goal is to divide a concept into smaller concepts so that the algorithm can learn easily. Inspired by these two similar ideas, in our work we focus on semantic exploration and decomposition of the dataset through anchors.

4.3 AnchorViz

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

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

DO2. Spread the dataset based on concepts.

DO3. Show how user-defined concepts impact item positions.

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

DO5. Optimize for efficient reviewing process.

4.3.1 *Interface and system design*

The interface for AnchorViz (Figure 4.1) contains a RadViz-based visualization that shows both labeled and unlabeled items (Figure 4.1C) as well as anchors that represent concepts (Figure 4.1D). When users click an item, the thumbnail view (Figure 4.1B) switches to show its content, and users can inspect the item to provide a label. This gives users a quick way to check if the item is an error and provide a correct label (DO5).

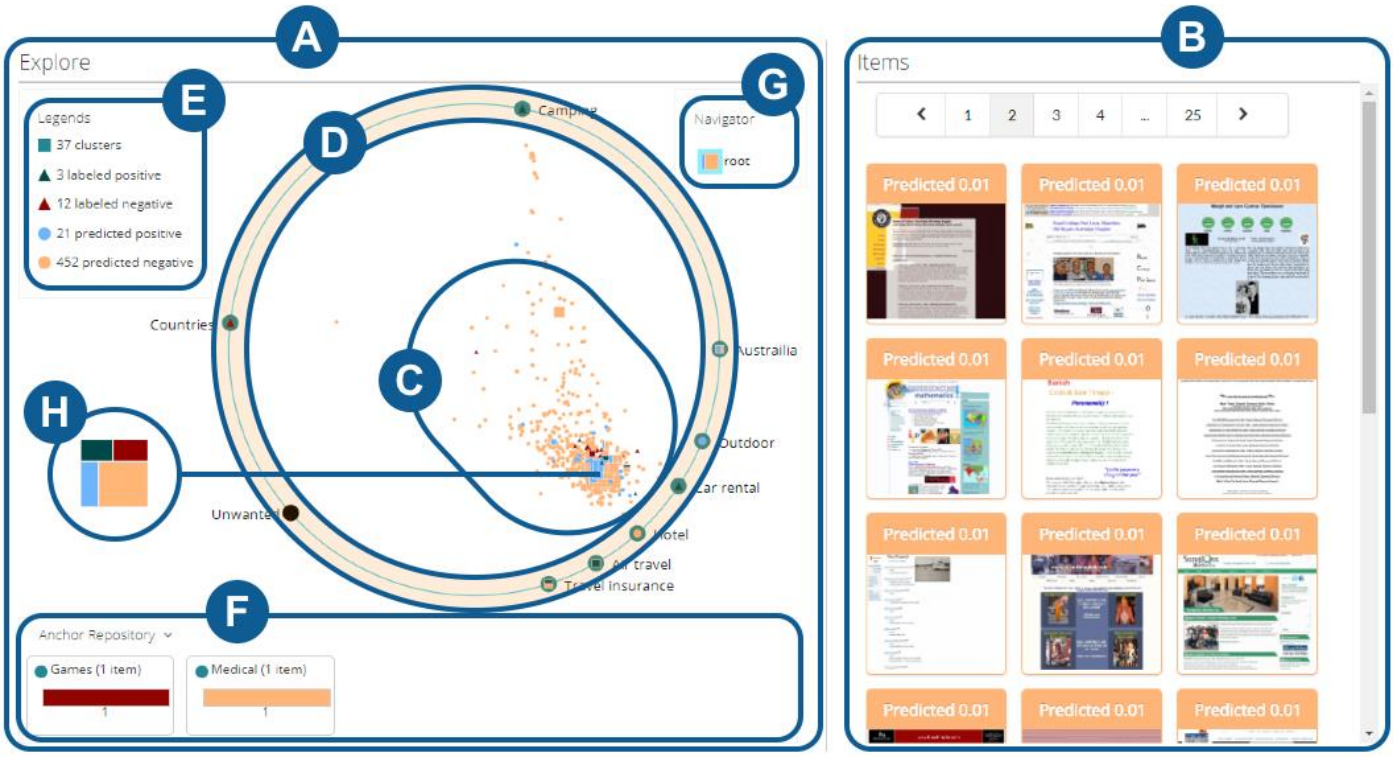

Figure 4.1: Overview of AnchorViz interface. The interface has an Explore pane (A) that includes the visualization and an Items pane (B) which shows a paginated grid of thumbnails of all currently visible items in the left pane. The visualization shows data points (C) within the outer circle (D) where anchors are positioned. The legend for data points (E) also acts as filters. The Anchor Repository (F) contains unused anchors. The Navigator (G) shows which cluster the visualization is displaying in the current navigation stack. Clusters are represented as treemap-style squares (H).

4.3.1.1 Define concepts with example-based anchors

In this work, we let users define a concept via examples (DO1), which is a common view of how a concept is stored in the brain [Nosofsky 1986]. Thus, these anchors are "example-based" anchors. We leave exploring other types of anchors as future work. To create an anchor, users can drag an item to the outer circle (Figure 4.1D). Users can also drag an item to an existing anchor to modify the corresponding concept. When a user clicks on an anchor to reveal the anchor detail view, users can provide a meaningful name for the anchor, view the list of items that belong to the anchor, and remove items that no longer belong to the anchor. In addition, users can hide an anchor from the visualization. The hidden anchors sit inside the "Anchor Repository" at the bottom drawer of the left pane (Figure 4.1F) where users can restore the anchors to the visualization.

4.3.1.2 Manipulate topology and layout

Once a concept is defined, the user should be able to see the correlation between the concept and the items (DO2; e.g., webpage A is more like "catering") as well as the relationship between several concepts with respect to the items (DO3; e.g., webpage B is more like "catering" and less like "restaurant menu," while webpage C is unrelated to both concepts). We map the relative similarity of the items to the concepts to the position of items in a non-orthogonal coordinate system in a circle; the center point of the outer circle to each anchor forms a set of axes on a 2D surface. Namely, an axis k is a vector with the length of the outer circle's radius r and an angle θ to the corresponding anchor (Eq 4.1).

$$\vec{V}_k = (r \cdot cos\theta, r \cdot sin\theta) \quad \textbf{(4.1)}$$

An item along an axis forms a vector with an angle identical to that of the axis and a magnitude of the cosine similarity in the BoW space between the item and the items in the anchor. The final position of an item in the visualization is the sum of the vectors to each anchor (Eq 4.2).

$$\boldsymbol{Position(i) = (x_i, y_i) = \sum_k \frac{value_k(i)}{\sum_j value_j(i)} \cdot \vec{V}_k} \quad \textbf{(4.2)}$$

Thus, the items that are closer to an anchor are more similar to the items in the anchor. The items that are affected by moving the anchor will move along with it, whereas items that do not share any similarity will remain unmoved (DO3). In addition, since we want to ensure all items sit within the outer circle, an item's value on an axis is normalized by the sum of all its values on all the axes. This normalization follows the typical normalization used in RadViz [Hoffman 1999]. In this way, the users are effectively creating a topology in the semantic space defined by the anchors. We considered multiple visualization techniques for defining this topology, but ultimately chose RadViz because of its support for an arbitrary number of axes and its flexibility in positioning the axes while preserving the relative independence of the axes [Sharko et al. 2008]; we leave other visualization options for future work.

4.3.1.3 Contrast model predictions with concepts

Semantic data exploration alone does not address our goal of finding errors in a classifier if the model is not involved in the exploration process. By surfacing the model predictions and the labels on the semantic topology created by anchors, the user should be able to contrast the predictions with concepts (DO4) to look for potential inconsistencies in two different ways. First,

the user can look for items that are predicted to be in the opposite class as the class expected to be close to an anchor. For example, restaurant menus should be treated as a negative concept for the recipe class. If an item near a restaurant menu anchor is predicted as positive, then the item is a potential prediction error. Second, if an item is predicted or labeled as positive in a sea of negatively predicted or labeled items, that item is an outlier worth inspecting. We encode model predictions and user labels into color and shape of items as illustrated in Figure 4.1E. Users can also click on the categories in the legend to filter items based on their types.

4.3.1.4 Inspect groups of items

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

In our clustering algorithm, the distance between any two items is their cosine similarity in the BoW space. The distance between clusters (C_a and C_b), including single point clusters, is the average distance between pairs among two clusters (Eq 4.3).

$$\boldsymbol{distance(C_a, C_b) = \frac{\sum_i \sum_j distance(C_{ai}, C_{bj})}{|C_a||C_b|}} \quad \textbf{(4.3)}$$

In each HC step, the algorithm selects a pair with the shortest distance and merges the two as a new cluster. The step is repeated until there is only one cluster left. We reorganize the result

of HC, which is a binary tree, into an *n*-ary tree from the top tree node and expand the tree in a breath-first order. At any given node of the *n*-ary tree, it can contain leaves (items) or sub-trees (clusters) which can further divide into sub-trees and leaves. For our user study, we used $n = 500$ which is neither too cluttered nor too sparse. We leave choosing an optimal value of *n* for future work.

These pre-processed groups of similar items (hereafter, we refer to a group of similar items as a cluster), are displayed as treemap-style squares (Figure 4.1H) in the visualization; the size of the square is a function of the number of items inside the cluster. Namely, each square is also a single-level treemap [Shneiderman 1992] representing the composition of items in four categories (labeled positive, labeled negative, predicted positive, and predicted negative). With treemaps, we aim to provide an at-a-glance understanding of the distribution of items within so that users can make a quick decision to navigate into the cluster. For example, if the cluster contains predicted negative items and labeled positive items, there may be potential inconsistencies between the clustering, the classifier, or the labels that need to be investigated and resolved. We display the same treemap for each anchor as a way to visualize their item composition as well as their class association along with the descriptive name.

Each square representing a cluster is positioned in the visualization based on the average similarity of all the leaf items inside the cluster. Clusters move along with the anchors to which they are most similar, just like any individual items. When users click on a cluster, they navigate into the cluster and the visualization updates to show the items and clusters that belong to the cluster. The navigator on the top right (Figure 4.1F) shows which cluster the visualization is displaying in the current navigation stack. The “root” cluster at the top of the stack represents the

entire dataset. Users navigate back to a previous level through the navigator or double click on the white space in the Explore pane.

4.3.1.5 Implementation

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

4.4 Evaluation

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

4.4.1 *User study*

4.4.1.1 Participants

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

We categorized our participants into four groups according to their ML background. Four participants had taken courses in ML (P1–4), four occasionally built ML models in practice (P5–8), three frequently built ML models (P9–11), and five had a doctoral degree in ML-related fields (P12–16). Our participants' professional roles in the company were that of applied data scientist (9), program manager (2), researcher (1), financial manager (1), and software engineer (3). Six participants had a doctoral or professional degree, nine had a master's degree, and one had a bachelor's degree.

4.4.1.2 Background scenario and tasks

To evaluate the visualization, we provided the participants with a specific scenario. The premise for the user study was that the participant was developing a binary classifier using an iML tool (i.e., iteratively sampling for items, labeling the items, adding features, and debugging the target classifier). Through this iterative process, the participant had achieved almost 100% accuracy on the training set. Despite high training accuracy, the classifier performed poorly on a

held-out test set, and the participant could not recall any items or features where the classifier could be making a mistake. Therefore, the participant switched to exploring a large unlabeled dataset to find potential sources of errors.

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

4.4.1.3 Dataset, classifier, and setup

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

To control for the ratio between the labeled set and the unlabeled set, we uniformly sampled 4000 items from the full dataset (50% positive for cooking) and 400 items in the labeled items such that approximately 8–10% of the dataset was labeled. To simulate blindness to concepts within positive and negative classes of cooking, we removed 25 features related to cooking (e.g., appliances, seasoning), and we left only one n-gram in each feature to degrade its test accuracy further.

After training the classifier, we subsequently removed incorrect items in order to achieve 100% training accuracy. The final cooking classifier had 309 labeled items (130 positive), 12 features, 100% training accuracy, and 75.8% test accuracy. For participants to practice, we also picked a travel-related binary classifier to use throughout the tutorial. This classifier only appeared during the practice round.

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

4.4.1.4 Procedure

We conducted our study through video conferencing where participants shared their screens and thoughts aloud during the study, and we audio and screen-recorded the entire session. The user study consisted of four parts: The first part (20 minutes) involved an introduction to basic

ML knowledge such as classification, errors, precision and recall, description of the data set, overview of the study interface, and introduction of the “travel” class (webpages about traveling and tourism).

The second part (10–20 minutes) was a practice round using the travel class as a reference to get familiar with the interface, followed by an introduction to the “cooking” class (webpages about home cooking) which was used for the actual task. The third part was the actual task (20 minutes) where we asked the participants to use AnchorViz to find and label a diverse set of items where the participants disagree with the classifier’s predictions. At the end of the study (5+ minutes), we asked participants to complete a quick survey to assess their satisfaction and collect open-ended feedback. Each study was a 70 to 90-minute session with a $30 gift card compensation.

4.4.2 *Quantitative and qualitative data analysis*

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

4.4.2.1 Qualitative coding

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

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

4.4.2.2 Error categorization and generalization

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

We also measured the quality of items by evaluating a classifier trained with the discovered items against a held-out test set. There are several challenges here. One is in simulating a human's featuring ability which is required in the iML loop that we operate in. Another is that the test set may not include a representative item for a concept that the participant discovered. For example, there is not a single item with reference to "insect recipes" in the test set, but participants can find this concept during exploration of the unlabeled set. Fully acknowledging the challenges in evaluating the classifier's generalization performance, we computed the

classifier's accuracy on the held-out test set using the original feature set (37 features). Our justification is that comparing two classifiers with or without the discovered items on a fixed feature set would provide us with a glimpse into the quality of the items added.

4.4.2.3 Anchor effectiveness

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

Before further defining AEP and AER, we first define "contrasted items". Contrasted items aim to capture items that become salient through anchors. The steps to determine whether an item is a contrasted item are:

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

Then we define AEP and AER as

$$AEP = \frac{\#\ true\ errors\ in\ contrasted\ items}{\#\ contrasted\ items} \tag{4.4}$$

$$AER = \frac{\#\ true\ errors\ in\ contrasted\ items}{total\ \#\ true\ errors}. \tag{4.5}$$

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

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

4.5 Results

4.5.1 *Discovered items*

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

4.5.1.1 Error analysis

On average, participants discovered 40.9 errors (SD=19.6). Because of the high variability in the number of discovered items, we looked at the percentage of errors in the total number of discovered items as opposed to the count of errors. We also looked at the first 50 (mean number of items for PhD in ML group) items returned from algorithmic samplers. Of the unlabeled items (3691 items), the initial cooking classifier made prediction errors on 943 or 25.5% of the items and feature blindness errors on 886 or 24.0% of the items. Except for P10, who discovered errors at a rate of 24%, all participants discovered errors at a higher rate than the total error distribution, random, stratified random, or uncertainty sampler. The uncertainty sampler selected errors at 47.8% but only 8.7% of the sampled items were feature blindness errors. Again, except for P10, all participants discovered feature blindness errors at a rate higher than any algorithmic samplers. Figure 4.2 shows the distribution of errors among the discovered items across participants.

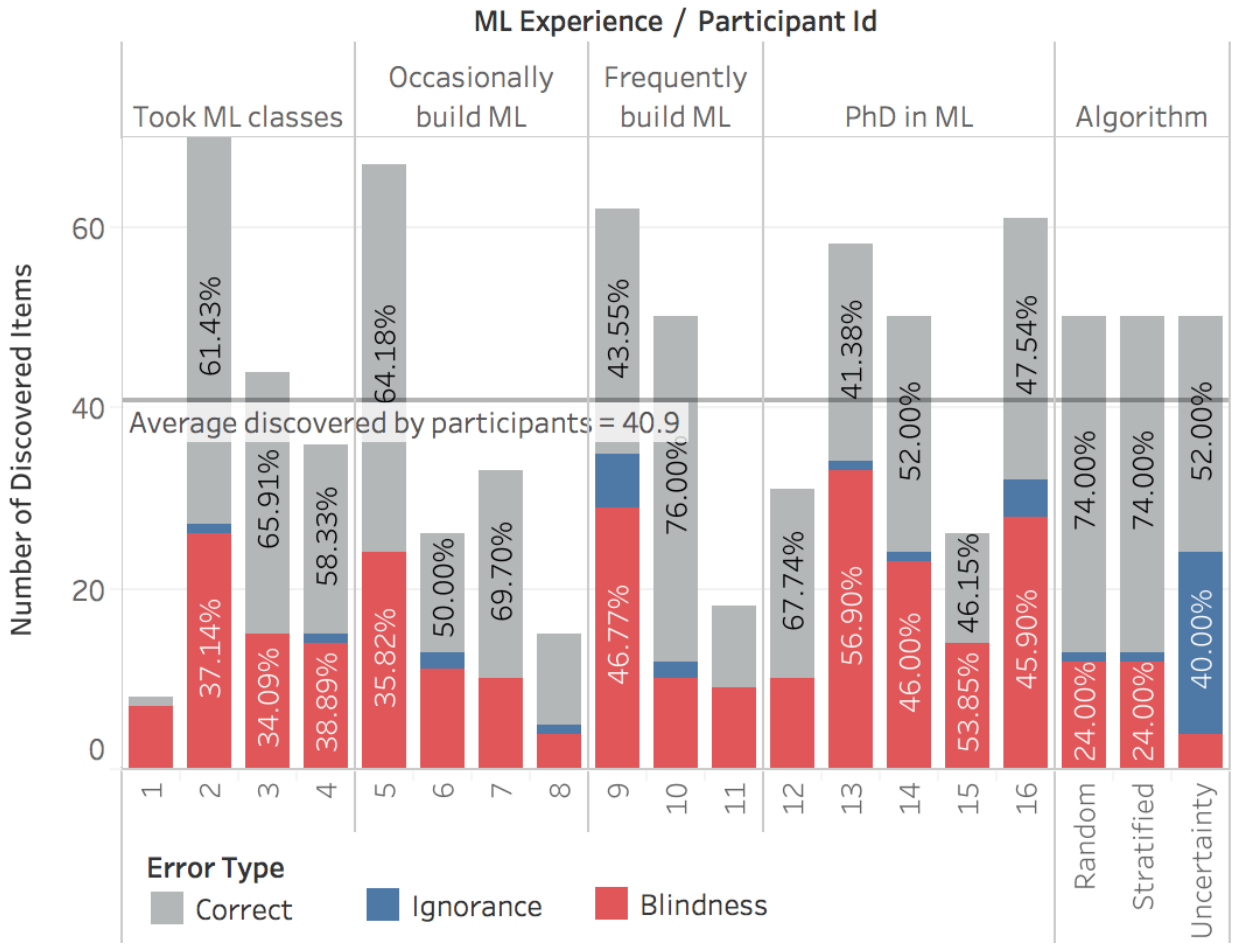

Figure 4.2: Distribution of errors across participants and algorithmic samplers. Text in each bar shows the percentage of errors in different categories. All participants except for P8 discovered feature blindness errors at a rate higher than any algorithmic samplers.

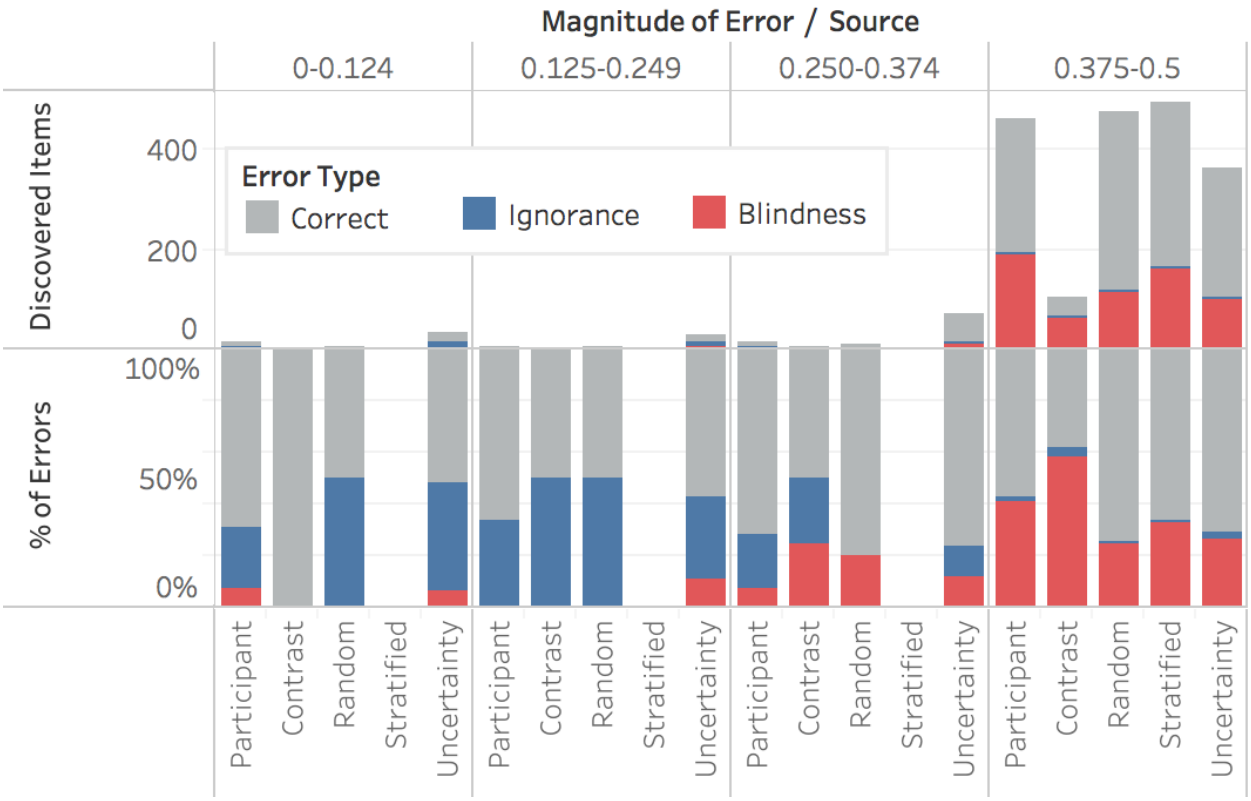

Figure 4.3: Distribution of discovered items (top) and magnitude of errors (bottom) across participants and algorithmic samplers. The items are skewed towards high magnitude errors, but items discovered by participants and contrasted items have a higher chance of getting high magnitude items than algorithmic samplers.

Figure 4.3 shows the distribution of the magnitude of errors across different samplers. Here, we looked at all the unique items discovered by the participants (n=493) and contrasted items among the discovered (n=111), and we selected the first 493 items from each of the algorithmic samplers. According to the distribution of discovered items, the dataset is skewed towards high confidence scores and a high magnitude of error. However, the percentage of errors indicates that the participants and contrasted group had a higher chance of discovering errors than algorithmic samplers. As expected, the uncertainty sampler selects more ignorance errors than any other samplers.

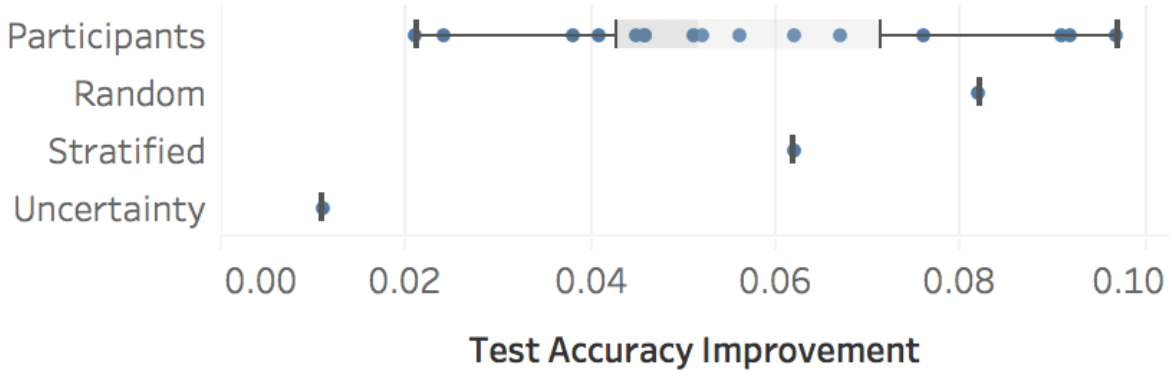

Figure 4.4: Test accuracy improvements across participants and algorithmic samplers. All participants built better performing classifiers than the uncertainty sampler, and three participants built better performing classifiers than random samplers. While random samplers performed well on the test set, they discover fewer feature blindness errors as indicated by Figure 4.3.

4.5.1.2 Test accuracy

Assuming that the features were fixed to the original set of 37 features, we compared the classifier accuracy on the held-out test set before and after the discovered items were added to the training set. As before, we selected the first 50 items from algorithmic samplers for comparison. Figure 4.4 shows that all participants made test accuracy improvements, and their classifiers performed better than the uncertainty sampler. Because the dataset was skewed towards high confidence scores, random samplers were able to select high magnitude error items and improved test accuracy. However, random samplers discovered fewer feature blindness errors as indicated by Figure 4.3, and three of our participants (P9, P13, P16) had classifiers with higher test accuracy than those created by random samplers.

4.5.2 *Anchor effectiveness*

Figure 4.5 shows the average AEP and AER of each participant. The reference line "Random" of AEP shows the baseline for AEP to be 0.255, the error rate in the dataset. This

baseline is how likely an item is an error if randomly selected by a user. As the figure shows, 12 out of 16 participants had an average AEP greater than random. This indicates that contrasted items are likely to bring errors to attention.

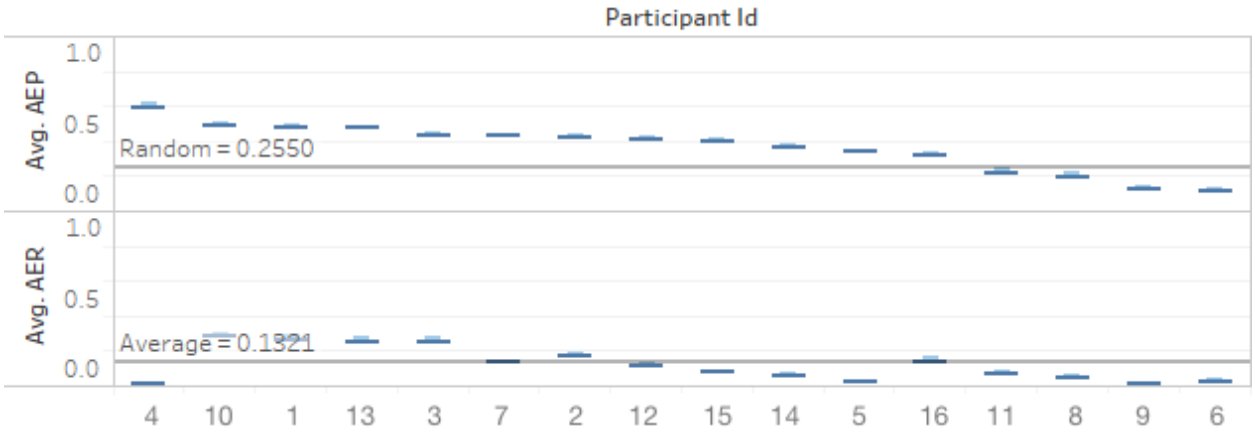

Figure 4.5: Average anchor effectiveness metrics (AEP/AER) for each participant. Most participants had better AEP than random.

For AER, we plotted the average across participants as a reference line since the same analogy in random is 1 (all the errors are in the dataset). The average AER (= 0.1321) indicates that only about 13% of the errors were the contrasted items. However, the goal of our tool is not to find all errors, but to find those errors that are critical. We further analyzed the types of errors in contrasted items and found that most errors in contrasted items had a higher chance of finding feature blindness errors than other approaches, including the algorithmic samplers, as we described in the error analysis section.

4.5.3 *User behavior*

In this section, we outline different exploration strategies that the participants used when interacting with the visualization tool from qualitative analysis. We focus on three key aspects of

the participants' use: (1) their reasons for creating anchors, (2) their exploration strategies, and (3) their understanding of classifier performance.

4.5.3.1 Reasons for creating anchors

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

4.5.3.2 Exploration strategies

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

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

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

Participants used the anchors to pull or push items of a specific prediction class, label class, or concept from the cloud of items. As mentioned earlier, some participants used the anchors to validate that there were a lot of non-English webpages in the dataset. Some participants used the labeled items as a way to validate that the anchors were pulling correct items. Some participants

moved the anchors to create a parallax effect to see which items were impacted by the anchor or to determine the effectiveness of anchors they created. Most of the time, the participants used the anchors to look for items near the anchors that were predicted in the opposite class from the labels of the items in the anchors. As participants interacted with anchors, they refined the anchors to make them more effective. P11 added similar items to an existing anchor to make the anchor stronger. P9 and P11 removed items from an anchor because the definition of the anchor had deviated from its initial intent. P4 renamed an anchor from "cucumber" to "vegetables" because the anchor's representative concept evolved over time. Most participants added items of similar concepts to existing anchors. When the participants discovered an item and they could not decide on its label, they added the item to an anchor as a way to collect similar items and to defer the decision until a later point (P12 - "*Cookbook can be negative in my opinion, but I'm not sure. It's about cooking and maybe it's related to cooking. Let's treat it as positive for now*.").

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

Some participants developed a repeatable process for exploration. P11's process was to find negatives around positive anchors, look at the clusters of mixed predictions near positive anchors, create appropriate anchors, and to repeat the steps. When reaching a saturation point for

exploration in the current view, P9 went back to the root cluster to start over, and P3 removed anchors to reset the topology.

4.5.4 *Classifier performance*

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

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

4.6 Discussion and Future Work

4.6.1 *Concept evolution and feature discovery*

Our evaluation reveals that AnchorViz not only helps users to find prediction errors, but also supports them in performing other necessary classifier building tasks. In particular, we observed that the participants used the anchors to facilitate concept evolution, which is "the labeler's process of defining and refining a concept in their minds [Lakkaraju et al. 2017]." Several participants used the anchors to collect items that they were unsure about (e.g., cookbooks, recipe links), deferred the decision for class assignment, and bulk-labeled the items at a later point. P13 said the interface is "*great for concept discovery and dynamic labeling.*" This interface can also be used for managing discovered concepts and "*creat[ing] new concepts that can be shared across people (P12)*," and for "*hierarchically expressible classification tasks (P16)*." In contrast to previous approaches to find unknown unknowns, our visualization also facilitates feature blindness discovery: P2 commented that this interface is useful for "*finding features I didn't know about and finding instances that can represent some useful concepts*." P13 said, "*it was also really useful for discovering items that were out of scope (e.g., foreign language).*"

4.6.2 *Anchor types and manipulation*

In our current design, we explore the use of example items and determine their similarity in the BoW space. However, creating an anchor with example items is only one way to define its corresponding concept. We envision using other types of anchors and distance metrics in AnchorViz. For example, instead of creating anchors based on items that users encounter, the

tool can support search to enable direct access to specific items, and create an anchor based on all items with the given search terms. This is similar to P10's suggestion: "*keyword search through the data to find tougher items from intuition and see where they lie on the interface*." In addition, the tool can let users specify a set of keywords to compute prevalence of the keywords or allow defining any arbitrary function that outputs a normalized score given any item. Furthermore, the current design treats all anchors equally. It is possible to add weights to anchors so that users can express their levels of interest on different concepts. We leave all these directions for future research.

4.6.3 *Integration into interactive model building loop*

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

Once an error is discovered through the tool, labeled and added to the training set, a new model reclassifies the dataset; the user continues to select the next item if there are no additional training errors, or addresses new training errors by modifying features or adding new labeled items. For example, a positive label on a webpage about an edible flower recipe is classified as negative even after retraining a model if the model does not have the appropriate feature representation for it (i.e., feature blindness error). The system prompts the user to provide a feature to fix the error, and upon adding an "edible flower" feature, the model is finally retrained

to correctly classify the webpage as positive. In addition, the tool could be useful for bulk labeling of items within an anchor or items co-located in a topology created by many anchors. Since the tool supports capturing concepts in the form of anchors, another use of the tool is for the evaluation of the model performance at the concept level by looking at the error rate within the anchor or around the anchor.

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

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

4.6.4 *Interface improvements*

When we asked the participants what they liked about the interface, participants commented that the interface supported exploration in an intuitive and fun way. P4 said, "*visual inspection of outliers is intuitive*," and P6 said that the interface is "*very intuitive when the anchors are meaningful*." P13 liked the "*visualization and self-organization of item space.*" P14 commented during the study, "*I play too happily with it!*"

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

4.6.5 *Further evaluation*

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

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

4.7 Conclusion

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

4.8 Acknowledgments

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

Chapter 5. Toward a Framework for Artificial Intelligence Workflows

5.1 Introduction

In the previous two chapters, we focused on designing two VA tools for AI system development. In order to identify other ways in which VA tools may be helpful in AI development, we talked with more AI developers who worked on a more diverse set of projects and models. We found that the lack of a shared language to describe AI development workflows made it difficult for us as (a VA researcher and designer) to come up with a solution that could fit everyone's needs. Thus, we decided to take a step back to understand the big picture of AI development, and identify the common challenges in the workflows as opportunities for future VA design.

To gather a more complete picture of the overall AI development process, we conducted a series of on-site contextual interviews at an AI research institute to understand the current practices of AI development and the issues AI developers encounter. During the study, we interviewed developers who work on a diverse set of AI systems, such as question answering, diagram parsing, citation recommendation, and entity extraction. These developers had used both ML and non-ML methods to develop AI systems. Based on the results of the study, we propose a framework to illustrate the different stages in AI development, highlighting the tasks and challenges developers must address. Based on the findings, we derive a set of implications for designers and researchers interested in creating tools for AI system development.

The contributions of this chapter are threefold: First, the empirical study on current AI development processes lays the groundwork for designers to understand which parts of the various workflows their tools aim to support. Second, the challenges extracted from the

fieldwork shed light on potential opportunities for research. Finally, the implications have the potential to guide the design of future AI development tools.

This chapter is collaborative work with Kyle Lo and my advisor Cecilia Aragon. Kyle is an applied scientist in AI2 who has a solid background and fruitful experiences in AI development. I did the whole interview study and analysis, and proposed the framework and its associated challenges. Kyle helped sharpen and clarify many AI related concepts and worked on writing with me. Cecilia helped further polish the writing and positioned my research in context as a framework. A shortened version of this chapter is under review for CHI 2020.

In this chapter, we address my thesis statement by extracting the key elements and tasks from the AI developers we studied to come up with a framework to describe AI development workflows. In this we aim to lay the groundwork for future VA researchers and designers to position their VA design within a clearer context of AI development since currently there is no shared language for describing such a workflow. The identified challenges are also good starting points from which to further propose VA tools to solve these challenges, fostering future VA design and research.

5.2 Related Work

Past research has focused on understanding current practices in AI and data science. For example, Kandel et al. conducted an interview study with 35 data analysts from industry to extract their needs and barriers in adopting visualization tools [Kandel et al. 2012]. Muller et al. interviewed 21 data science practitioners to understand how they work with data [Muller et al. 2019]. In more recent years, the call for fairness, transparency, and accountability in AI systems has fostered more studies on how ML developers understand their models [Hohman et al. 2019]

and what is needed to improve model fairness [Holstein et al. 2019]. All of the above-mentioned works examine certain aspects of the AI development process, yet none focus on capturing the entire process, nor do they highlight issues from workflow perspectives. In addition, data science and AI are not equivalent. Even though sometimes data scientists may build AI systems to handle data-related tasks, the goals of the two fields are still slightly different: data science focuses more on getting insights from data, whereas AI focuses on building systems that can perform human tasks. Since most of the past studies target data science workflows, a more thorough examination of AI workflows is needed, which is the goal of this study.

Another related direction is Yang et al.'s work on how non-experts build ML models [Yang et al. 2018]. In their study, Yang et al. survey 98 non-experts and interview 14 non-expert ML model builders as well as 10 ML consultants to extract their development process and the required knowledge, highlighting the challenges and misconceptions that non-experts have in building models. Even though Yang et al.'s work focuses more on model development, their goals are to identify the mistakes and pitfalls non-experts may encounter since what they aim for is a new ML paradigm called *machine teaching* [Simard et al. 2017]. Machine teaching emphasizes how humans 'teach' machines: Namely, which processes and languages human teachers should use so that models/machines as students can more easily understand and learn what human teachers want them to learn. But since machine teaching is a relatively new paradigm with few practitioners, in this paper, we focus on the prevailing AI system development workflows, rather than machine teaching workflows.

Forsythe's ethnographic studies of AI system development in the 1990s yielded some excellent early discoveries [Forsythe 1993]. Although the field of AI has changed significantly in

the past 20 years, Forsythe's work highlights the fact that the construction of AI systems is a social process and AI systems are embedded with developers' values. Our work considers the challenges AI developers encounter during development as part of a socio-technical system, and thus our framework can provide a starting point for re-examining the social aspects of AI development to continue Forsythe's efforts.

As described in Chapter 2, there is a large amount of work in the VIS field focusing on designing visualization tools for machine learning tasks. Such work usually includes a user study and covers a portion of AI development workflows. However, most of this work does not include in-depth ethnographies nor does it specifically focus on systematically extracting challenges and illustrating complete development workflows. One more general example is Sacha et al.'s work on building an ontology for visual analytics (VA) assisted machine learning [Sacha et al. 2018]. They extract VA workflows from 21 published VA papers to develop an ontology with the goal of providing a theoretical foundation for VA-assisted ML. They also envision using the ontology to optimize model development workflows based on existing VA tool design. Although their work covers AI development workflows, it is based on existing VA work, which is second-hand and limited in scope, rather than first-hand empirical studies on the current AI development process as a whole. In the study of this chapter, we aim to go into more depth and reveal more AI developers' needs from a first-hand empirical study.

5.3 Background and Methods

5.3.1 *Research site and data collection*

Our interview study was conducted at a research institute in the United States for about eight months. We were on the field site a few days per week to conduct both formal interviews and informal interaction. I was also a participant observer as a student intern who previously built a visualization tool at the institute. Thirteen people participated in individual semi-structured interviews. Each interview session lasted about one hour.

During the semi-structured interviews, we asked the participants to describe their projects, models, and inputs and outputs of their models. Then we focused on their development workflows, in particular how they design and analyze their systems as well as challenges they face. Example interview questions are

1. Could you give me some background on your projects?
    a. What are the typical inputs/outputs?
    b. What packages/environments do you use?
2. Could you use one recent example to explain your typical workflows to develop your system?
3. Are there typical metrics / ways of evaluation you use?
4. Are there parameters that you usually tune?

Two out of the 13 participants were female, and the rest were male. Nine participants held PhD degrees in AI-related fields, one participant had an MS degree in Statistics, and the remaining three participants were PhD students interning in the institute. All the participants worked on their own projects, but some were collaborating on the same projects. The AI fields

they focused on ranged from natural language processing (NLP), to question answering (QA), to computer vision. Table 2 shows participant IDs and their background.

Table 2: Background of interview participants

#	Field	Gender	Degree
P1	NLP	Male	PhD
P2	Question Answering	Male	PhD
P3	Vision	Male	PhD
P4	Question Answering	Female	PhD
P5	NLP	Male	MS
P6	NLP	Male	PhD
P7	Question Answering	Male	PhD Student/Intern
P8	NLP	Male	PhD
P9	NLP	Male	PhD
P10	NLP	Male	PhD
P11	NLP	Male	PhD
P12	Vision	Female	PhD Student/Intern
P13	Question Answering	Male	PhD Student/Intern

5.3.2 *Data analysis*

All the interviews were recorded and later transcribed. Then we conducted a thematic analysis [Braun and Clarke 2006] on all the interview transcripts: we first reviewed each transcript, open-coded it, and pulled out quotes related to AI development workflows. The extracted quotes were collected in a spreadsheet, and we performed a second round of coding to categorize the quotes into background, issues, workflow, and other notes as well as the codes under each category as shown in Table 3. For deeper analysis, we further grouped the issues into themes as shown in Table 4.

Table 3. Categories and corresponding codes used in second round of coding

Category	Code	Definition
Background	Dataset	Datasets used in their projects
Background	Input	Inputs of a model
Background	Metrics	Metrics used in analyses
Background	Model	Models used in a project
Background	Output	Model outputs
Background	Parameter	Model parameters
Background	Project	Project of AI developers
Background	Tool	Tools used by AI developers
Issues	Issue	Issues encountered by AI developers
Issues	Need	Needs mentioned by AI developers
Notes	Notes	Additional information that may be useful (for my analysis)
Workflow	Criteria	Standards used by AI developers during their analysis
Workflow	Goal	Goal of a project / Goal of a task
Workflow	Hypothesis	Any hypothesis the AI developers came up with
Workflow	Presentation	How the AI developers present their results/findings
Workflow	Process	Any development process mentioned by AI developers
Workflow	Reference	Information from other places (e.g., other papers/models) that the AI developers used as references
Workflow	Strategy	Strategies they used during development
Workflow	Task	Tasks in AI development workflows
Workflow	Variation	Any variation (e.g., other datasets) the AI developers may try in development

Table 4. Theme codes for analyzing issues encountered by AI developers

Theme Code	**Details**
Explorative & Flexible	As the process of development is exploratory, it is not easy to define structures at beginning and provide supports
Indirect	Impact of the changes is indirect; not intuitive; developers have limited control
Time-Value Trade-off	Spending too much time on one thing may yield little benefit, even though it will lead to a better understanding
Time/Resource Limitation	Limited time and/or resources
Tool limitation	Sometimes, they are constrained to what their tools can provide
Prone to Unnoticeable Mistakes	Very easy to make mistakes that are not easy to notice
Forget	There are many changes/variables/parameters so it’s easy to forget what they mean or what changes are made
Inter-related	Components are inter-related. It’s hard to figure out which part is causing the problem. It's not easy to find the best solution as many parts must be considered together.
Nondeterministic	Nondeterministic error; got lucky
Manually	Things have to be done manually
Not Intuitive	Not intuitive
Challenges of Going Forward	It’s not easy to figure out what to do next
Not Fully Understand	May not fully understand the tool/model/algorithm
Dig into Details	Need to dig into details for analysis
Dependency	Dependency-related issues

After two rounds of coding, we used affinity diagramming to synthesize the quotes into a framework (Figure 5.1) to suggest a framework for AI development workflows. The process of creating the framework relies on inductive reasoning since we derived the framework based on the data we collected. The initial framework and themes of issues were examined by my collaborator Kyle, who is also an AI developer. We worked together to iterate to a version we mutually agreed on.

There are four primary stages in this framework: (A) Data Processing, (B) Initial Development, (C) Parameter Engineering, and (D) Design Evaluation and Analysis. We also highlighted key elements and tasks in each stage. In the next two sections, we first describe the framework, and then discuss issues in AI development workflows using the framework.

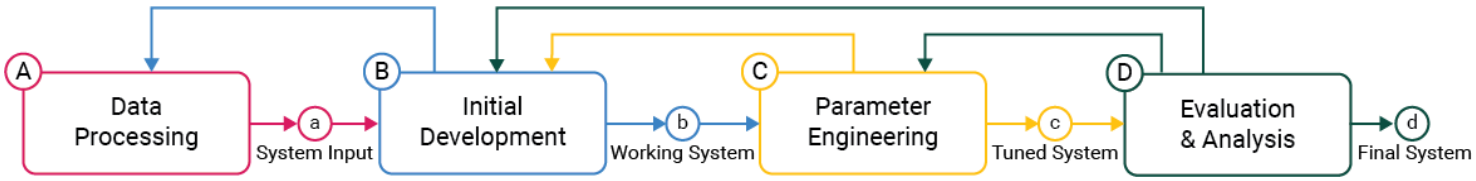

Figure 5.1: Overview of proposed framework for AI development workflows. The framework is constructed based on a thematic analysis of interview transcripts. The four stages are iterative. Each stage has different elements and tasks.

5.4 A Framework for AI Development Workflows

An AI development workflow can be roughly divided into four stages: (A) data processing, (B) initial development, (C) parameter engineering, and (D) evaluation and analysis. The bulk of the development workflow is spent in Stages B and C, in which developers conduct experiments to assess whether a particular AI system configuration works well on the curated data from Stage A. Stage B involves experimentation with different system architectures and training models, whereas Stage C involves experimentation with different parameters and learning algorithms on

promising system designs from Stage B. In this section, we first explain the notation used in the framework figures, and then illustrate each stage.

5.4.1 *Notation in the framework figures & terminology*

The four figures below illustrate the framework for AI workflows. Each corresponds to a stage. In these figures, we use circles to represent key elements in the stage; all the circles are numbered or labeled with letters. The four key elements between and after stages are labeled alphabetically. The arrows represent tasks that AI developers perform on these elements. Dashed arrows or circles are optional. Not all the cases will have corresponding tasks or elements. Squares indicate references, which include prior work and state-of-the-art performance. In addition to shapes, each stage is color-coded: Stage A is magenta, Stage B is sky blue, Stage C is yellow, and Stage D is dark green. If a shape is in a color that is different from the stage color, it originates in an earlier stage. We preserve the numbering from the original stage of such an element.

To make the illustration in the following sections clear, we define terminology here:

1. **AI developers**: When we mention "AI developers" in later sections of this chapter, we refer only to those AI developers who participated in the study. We do not intend to claim that our results cover all possible cases, but we also believe many of the findings can be generalized to other AI developers. For the sake of simplicity, we will use "AI developers" in our description, but will highlight the limitations of our work in Section 5.7.
2. **AI system vs. model**: We use "AI system" to refer to a system that performs any AI task, whereas a "model" refers to an ML model that can learn from data. An AI system can

consist of a single ML model or a combination of multiple ML models. An AI system can also be non-ML based (e.g., rule-based AI systems).

3. **Experiment**: We refer to an "experiment" as a single instance of providing a set of input data to an AI system and getting the outputs from the system. If an AI system is ML-based, then training may happen as part of the experiment. Some experiments may reuse trained models or do not have ML models, and thus would not include training.
4. **Log**: We refer to a "log" as outputs generated during the course of an experiment. This might include intermediate outputs for monitoring progress or debugging while the system is running, or final outputs (i.e., system predictions and evaluation metrics).
5. **Configuration**: We refer to a "configuration" as the collection of attributes or descriptors that distinguishes one instance of an AI system from another. For example, two AI systems with the same ML models but with different parameter settings (e.g., learning rate) are considered to have different configurations.
6. **Architecture**: We refer to an "architecture" as the structural design of an AI system. This includes pipeline and components of non-ML systems, model types of traditional MLs (e.g., using a decision forest model vs support vector machines (SVMs)), as well as the structure of NNs in deep learning (e.g., the number of neurons in each layer).
7. **Weights:** We use "weights" to refer to a set of values in an AI system that are derived from data. For example, in non-ML AI systems, an AI system can contain a knowledge graph that contains edge weights, which are determined based on the training data. In traditional ML such as SVMs, each feature has a weight after training. Similarly, in deep learning, each edge in an NN has a weight learned from data to decide how to use

information from the source node of the edges. Note that we do not directly use weights to refer to the configurable weights in ensemble methods unless they are also learned from data.

5.4.2 *Stage A: Data processing*

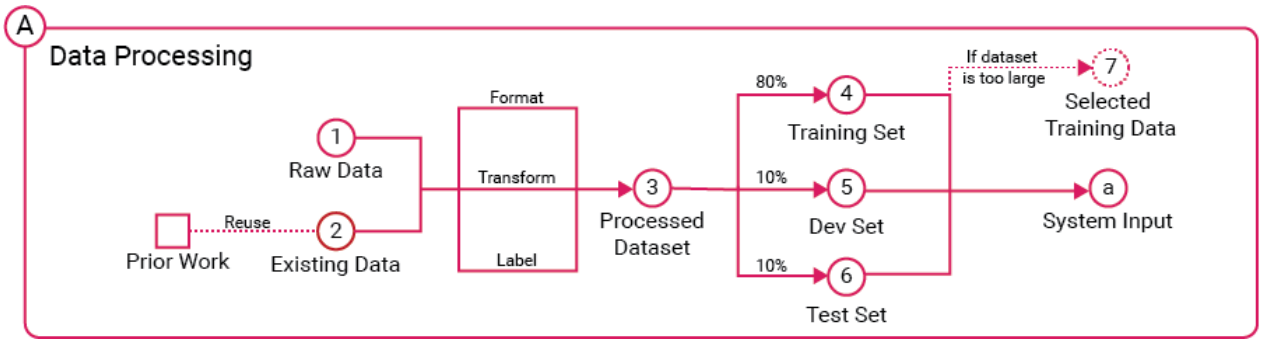

Figure 5.2: Stage A, Data Processing, contains data processing tasks. AI developers convert data from original formats into the AI system's input. The processed dataset is usually split into three sets (i.e., training, dev, and test) to prevent overfitting.

Typical AI development starts from data processing. The source of data can be either newly collected data (i.e., **(A1) raw data**) or **(A2) existing data.** Existing data may come from prior work of their own or from other people in the field. Regardless of the source, AI developers must process the data before input into the system. Types of processing range from simply formatting data, to transforming it (e.g., pretraining embeddings, extracting features, augmenting data), to labeling.

Small datasets can be saved as static files, while large datasets may require dynamic processing during model execution. In most ML packages (e.g., Tensorflow and PyTorch) as well as in self-built libraries, data processing is encapsulated in a data supplier module which handles loading, processing, and providing data to other AI system components such as models.

In deep learning, for NN models that train over multiple passes through the data (each pass is called an *epoch*), data suppliers must also handle data shuffling.

Once AI developers have the **(A3) processed dataset**, they partition the dataset into three parts: **(A4) training set**, **(A5) dev set** (short for development set, also called *validation set*), and **(A6) test set**. We refer to all three sets as **(a) system input**. The training set typically comprises a majority of the processed dataset (e.g., 80%), while the dev and test sets comprise just enough data to properly evaluate system performance (e.g., 10% each). The reason why AI developers need to have this division is due to the concern about *overfitting* (i.e., the learned model applies only to the learning data and thus does not work with real-world data) and *underfitting* (i.e., the learned model picks up some relationships in the learning data that are not sophisticated enough to describe real relationships and thus does not work with real-world data either) .

In ML, when a model overfits or underfits a dataset, it will have nearly perfect performance on that dataset but not on real-world data. Therefore, to prevent overfitting and underfitting, AI developers usually train on a training dataset, and then use performance on a separate test set to evaluate whether the model will generalize well to new data. However, if AI developers make decisions based on test set performance too many times, it can lead to another form of overfitting/underfitting, even though the model does not directly train on the test set. To prevent this, AI developers create a dev set that is used to assess performance and make design decisions during development, and leave the test set for evaluation at the end of the development cycle. Non-ML systems also use training-dev-test dataset splits. Although the training set is not used for learning, it is used for constructing the systems (e.g., extracted rules for a rule-based system). Therefore, the splits prevent similar overfitting/underfitting problems.

When the training set is large, AI developers select a small set of representative examples to use in development: this is the **(A7) selected training data** in Figure 5.2. Since this dataset is small, the turnaround time of an experiment is shorter. In subsequent experiments during Stages B and C, it is up to the developer's discretion whether it is appropriate to use A4 or A7. Smaller A7 leads to faster turnaround time for experiments but using less data can lead to worse-performing models and mistaken conclusions about which architectures/configurations are good.

5.4.3 *Stage B: Initial development*

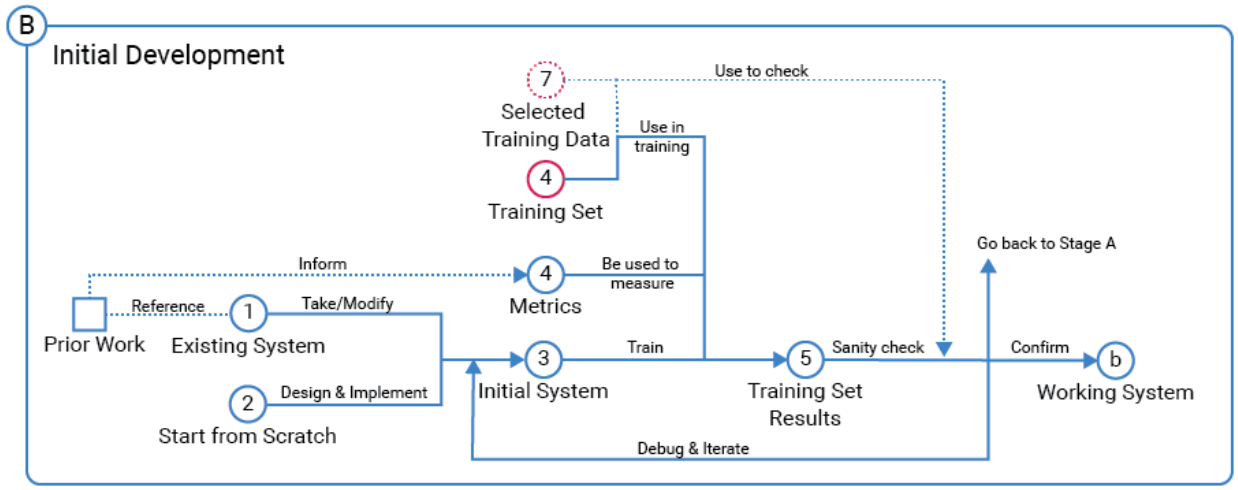

Figure 5.3: In Stage B, *Initial Development*, AI developers focus on getting an initial system to work. The initial system can be a modification of an existing system, or a newly designed system. AI developers may need to iterate a few times on the system code and do sanity checks with the training set to ensure the system is bug-free.

To start building an AI system, developers take one of two routes: modify an **(B1) existing system** or **(B2) start from scratch**. If an existing system tackles a similar problem, they may directly take or modify that system to fit their use case. In some cases, this might be as simple as modifying their data to fit the input format of the existing system. However, when the data or problem setting are sufficiently different from prior work, they may reuse the architecture but

need to reconfigure the system. If the system has ML models, then these models must be retrained to get new weights.

> *"If your learned weights should be super different from the AlexNet (a deep learning model for image recognition) pre-trained network then you may just train it from scratch." – Interviewee P3*

If reusing existing systems is not feasible, developers may opt to start over, designing and implementing their systems from scratch. This requires deciding on the kind of architectures and components to use. Each project may use different libraries to implement these designs. The choice of libraries depends on the traditions and trends in each research group. AI developers usually start from a simple system before they design any complex system, and iteratively add more components during development.

5.4.3.1 Pipeline vs. end-to-end systems

There are two major AI system architecture and component designs: *pipeline* and *end-to-end*. Pipeline systems are characterized by a collection of modular components that generate outputs for other components to take as inputs. For example, a QA system may have a language processing component for analyzing an input question, a knowledge extraction component to retrieve knowledge from a corpus or knowledge base, and a reasoning component that synthesizes the analyzed question and retrieved knowledge to formulate an answer. Each component can be a traditional feature-based ML model, a deep learning model, or a non-ML based component.

In pipeline systems, components are designed and tuned separately. The advantage of a pipeline system is that it is easier to isolate efforts to improve individual components. In addition, it is more convenient to reuse components of one pipeline across future systems. For example, word vectors are reusable in many systems instead of retraining from scratch each time; a knowledge retrieval module in QA system can be reused across other (even non-QA) systems if it is general enough. However, errors in upstream components propagate to downstream components. There is no way for a component in the later part of the pipeline to fix errors in an upstream component. Furthermore, improvements in components may be inconsistent. For example, improvements in upstream components may decrease the performance of their downstream components (e.g., Let C_U and C_D be the upstream and downstream components, respectively. Also let C_U make 2 types of errors Err_A and Err_B, of which Err_A is more common. C_D is optimized to anticipate error Err_A but does not handle Err_B well. Improving C_U by reducing Err_A significantly at the cost of slightly increasing Err_B might result in an overall decrease in performance due to how C_D was designed.).

In contrast, in end-to-end AI systems, which are typically implemented as deep-learning models, components are represented as intuitive groupings of computational units (e.g. layers of neurons). Training end-to-end means all components are simultaneously tuned to maximize their use for other components. For example, a QA system might benefit from end-to-end training to ensure that the question analysis component transforms the input question into a representation that is most useful for later synthesis. Therefore, end-to-end typically solves problems with propagated errors, as well as eliminates inconsistency between components. However, training end-to-end often has a higher computational cost since the more complex the full model is, the

trickier it is to get the model to train well. This is because it is harder to make a judgment about whether the model design is bad, or the model does not have the right set of parameters.

5.4.3.2 Start simple and iterate

In either setting, AI developers usually start from a simple model before they design any complex pipeline or architecture, and iteratively add more components during development.

> *"So in general I try to when I start working on a specific problem I try to define a very simple model, which takes the inputs and produces results as outputs. So at the first stage it's more of being able to read in the inputs and produce outputs in the required format. It's mostly data processing that goes into it. And I'll try to alternately add more and more features on it, making assumptions along the way about ... Yeah, you know, testing whether simplistic things work first and so that I can add more learning to it." – Interviewee P8*

Each project may use different libraries to implement these designs. When building NN models, some may use *Tensorflow*[1], some *PyTorch*[2], and some *Theano*[3]. When building traditional ML models, *Scikit-learn*[4] is a commonly used Python library. The choice of libraries depends on the traditions and trends in each research group.

---

[1] TensorFlow https://www.tensorflow.org/

[2] PyTorch https://pytorch.org/

[3] Theano http://deeplearning.net/software/theano/

[4] Scikit-learn https://scikit-learn.org

5.4.3.3 Sanity-check initial systems

Once AI developers have an **(B3) initial system**, they must periodically assess whether their system is working during iterative development. This step includes using the **(A4) training set** to train the models in the initial system and using **(B4) metrics** to measure the results. Some researchers define these metrics before the development process starts to avoid bias such as choosing metrics that favor the results. Other researchers are not very strict about this step, and they may decide which metrics they want to use after they get the **(B5) training set results**. No matter which attitude they hold, metrics are necessary for *sanity checks*. Usually, if there is prior work related to the problem they are working on, they will use metrics from previous papers; if there is no such work, or they find flaws in previous metrics, they may propose new metrics. Typical metrics include accuracy, loss, F1, precision, recall, and precision-recall curve (PR curve). There are also project or domain-dependent metrics. For instance, some AI systems consist of two models, and thus AI developers may need to combine metrics of the two models into a single metric.

To examine the metrics, they often look at a summary table of the metrics, or they use *Jupyter Notebook*[5] to plot metrics using *Matplotlib*[6]. Some also use tools such as *TensorBoard*[7] to examine metrics over epochs (a pass of a model over the training data is called an *epoch*). TensorBoard is a dashboard-based visualization tool provided by Google that loads logs from

---

[5] Jupyter Notebook https://jupyter.org/

[6] Matplotlib https://matplotlib.org/

[7] TensorBoard https://www.tensorflow.org/guide/summaries_and_tensorboard

both TensorFlow and PyTorch models. It provides a variety of ways to visualize the logs and allows online visualization (plotting while logs are still growing). By looking at the graphs, AI developers can make conclusions such as inferring there may be a bug:

> *"For example, after these two epochs, my accuracy is not going up then I will know that there's a bug." – Interviewee P13*

If AI developers have **(A7) selected training data** from Stage A, they may use them to check the system's results. This can be either training with the whole training set and then examining the selected examples in detail, or simply training on the selected training data and then determining whether the results look reasonable.

> *"From the very first experiment I run I usually have ... it may not be on the entire data set, but even if it's on the small data set itself I try to make sure that even on the small data set I'm actually evaluating in terms of the same metric that I would use." – Interviewee P8*

The goal of this step is to see whether the system is doing what they want it to do. For example, they want to check whether the model in the system is learning, or whether the model is overfitting or underfitting. In this step, some developers terminate the experiment early if they notice the system is not performing well or correctly.

> *"We see a few experiments that failed, but I intentionally made them fail because I wanted a specific sub-combination of the Cartesian product, so we made some of them fail fast." – Interviewee P1*

In this stage, AI developers do not usually run many experiments. They tend to focus on one experiment to see whether the idea works.

> *"In the initial stages I just run one experiment maybe. I don't have many experiments running. It's mostly because I'm testing the assumptions that I'm making about data set, or ways specific learning would work." – Interviewee P8*

If they find any issue with the system, they try to fix it, and then redo the experiment for another sanity check. Sometimes they may find issues with the data (e.g., incorrect labels, not enough training examples for certain classes, bugs in data suppliers, etc.), in which case they may go back to Stage A.

In this stage, AI developers may not spend much time tuning parameters; they save this effort until later in the parameter engineering stage. Once they are confident their models are working as expected, they will have a **(b) working system** and move to the next stage.

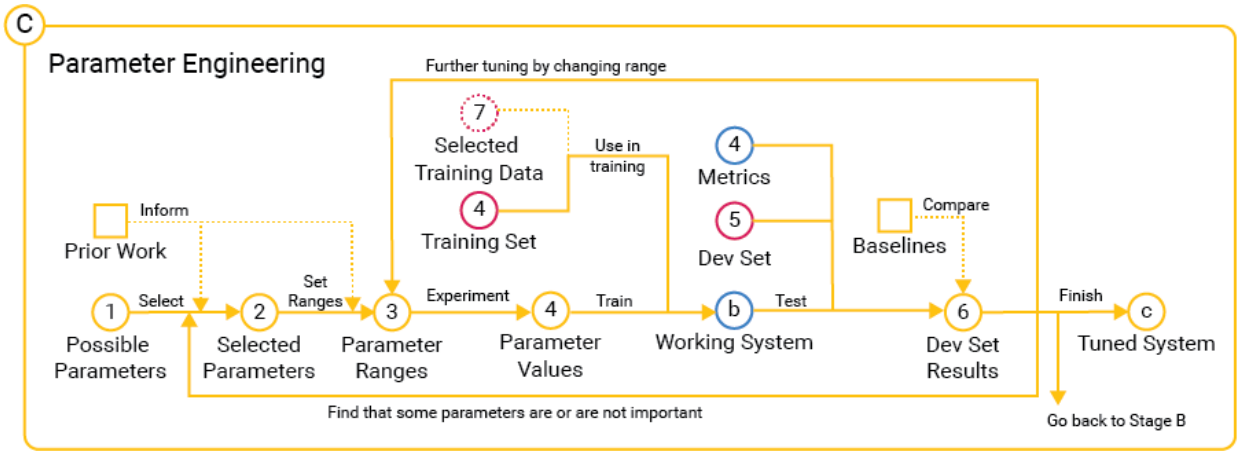

Figure 5.4: Stage C, *Parameter Engineering*, consists of two major tasks: parameter selection and parameter tuning. AI systems usually need a good set of parameter configurations to get decent performance. The two tasks are also iterative and are evaluated on the dev set. AI developers usually have some baselines to compare with when tuning the system.

5.4.4 *Stage C: Parameter engineering*

AI systems, especially those using ML models, can require very specific parameters to yield good performance. Therefore, in the parameter engineering stage, AI developers search the parameter space to find the setting that optimizes performance of their **(b) working system**. Here parameters refer to categorical or numeric configurations (e.g., stochastic gradient descent (SGD) vs. Adam). For NN models, common categorical configurations include choice of learning algorithms. Common numeric configurations of models include learning rate, batch size, and number of epochs, and other parameters of their learning algorithms. For traditional ML models, common categorical configurations include the choice of regulation methods; numeric configurations differ from model to model. For example, decision forests require tuning the number of decision trees and the maximum depths of the trees. SVMs require tuning the regularization parameter. For non-ML AI systems, there are also other parameters to tune (e.g., some traditional AI systems use *integer linear programming* (ILP), which has a number of coefficients as parameters to tune). Note that in our definition, parameter engineering does not involve changing the system design or architecture.

Some parameters are used in data suppliers. One common parameter is batch size. Batch size refers to how much data the data supplier should provide in each request; the choice of batch size can impact how systems perform over training iterations. Some systems are not sensitive to batch size, whereas other systems require a good balance between batch size and learning rate:

> *"So here in our experiments in our vision data set the batch size doesn't matter too much." – Interviewee P3*

"Finding a good balance for the batch size, for the learning rate, these kind of things are, I think the most important things." – Interviewee P12

To tune parameters, AI developers usually start by figuring out which parameters are important. Parameters that are not influential are set to a fixed value. In other words, AI developers must select parameters from **(C1) possible parameters** and tune on the **(C2) selected parameters**.

"One of the aims of the analysis view when we set out to build it was coming up with a way to sort of shine light on how important each of the different parameters weight given the results that were ..." – Interviewee P1

"Yeah sometimes I take one parameter for instance and optimize it. I see that it's pretty stable over different settings and so on, so I don't really focus much on that parameter. I don't optimize it every time in combination with the other parameters. I just keep it fixed and go to the rest." – Interviewee P12

For parameters that are important, AI developers try finer-grained **(C3) parameter ranges**. However, choosing the right parameter range is not an easy task since it can vary from dataset to dataset.

"Oh, how do I know [what the right range is]? No one knows. You just have to try. There are some accepted ranges, commonly used ranges that I'm aware of. But for a particular dataset, I don't know. I will just try out. I'm more interested in as more accepted ranges than trying on outliers upfront." – Interviewee P9

Sometimes, AI developers decide on the starting point of the search based on intuition, and then they keep experimenting with different values based on the results they see when changing the parameter values. It can also be an iterative process to increase and decrease the values.

> *"So, you have some intuition in terms of how to set this entailment parameter. So you start from 0.7 this range and then you will like okay maybe I can try something that is smaller so you go back to try these things." – Interviewee P7*

If using a previously-built system, AI developers may use the parameters from the prior work to set the range of values for searching, which can save time. Sometimes, AI developers may simply use the same set of parameter values from prior work if their system is a modified version of an existing system and they want to compare the two:

> *"I can't make an improvement to an existing system and claim that that improvement is significant enough if I'm not using the same hyperparameter settings that someone else has used" – Interviewee P8*

For systems that can finish within a few hours, AI developers usually use automatic fine-tuning tools such as *hyperopt*[8], or use algorithms like grid search to experiment with a set of possible values once they decide the ranges they want to try. For systems that take a long time to finish an experiment, for which grid search or automatic tools are not feasible, they must set values manually.

---

[8] Hyperopt: https://github.com/hyperopt/hyperopt

For every set of **(C4) parameter values**, AI developers must use these values to experiment with the **(b) working system**. When the training set is too large, AI developers use the **(A7) selected training data** since the turnaround time is shorter. If the results look good, they attempt to use the whole **(A4) training set** in the experiment to see if the system also performs well on the whole set.

> *"I tuned all of these parameters with that [small set]. Once I was confident enough in our small set, then I changed some important parameters on the, just a handful of them, like maybe five % of them I changed on a larger training set. But all the others they were just tried on a smaller set." – Interviewee P9*

To get a shorter turnaround time, AI developers may also limit the number of training iterations for ML models during parameter engineering, as they can get a sense of how the ML models perform within only a few epochs.

> *"I try ... Usually these models you can let them run for 100 epochs, but usually so far as I have seen within 50 epochs you get to know the trend. Even at 30th epoch, you know what the trend is going to look like." – Interviewee P9*

To assess how well the system performs with the chosen parameter values, AI developers use the previously determined **(B4) metrics** to examine **(A5) dev set** results. They compare against the dev set performance of a baseline system, which is often a state-of-the-art system or a simple system that the developer expects should perform reasonably well. For new or difficult tasks, developers may compare against a random-guess baseline. If the performance of their system against the baseline is not promising at all, they may go back to Stage B to change the

design or even start a new design. If they decide to continue tuning, they either adjust the parameter ranges or add or remove selected parameters.

> *"As it evolves sometimes some parameter that I didn't find important at the beginning became more important later on" – Interviewee P12*

In other words, parameter selection and parameter fine-tuning are iterative processes. At first, AI developers may not discover the importance of all parameters. Instead, they select a few parameters, focus on tuning them, and decide whether to keep tuning them or not. They may discover new parameter values to try when working with other parameters.

Once they are done with the tuning, they have the **(c) tuned system**. They then move onto the next stage to evaluate the system with all the **(a) system input** as well as with the full number of iterations.

5.4.5 *Stage D: Design evaluation and analysis*

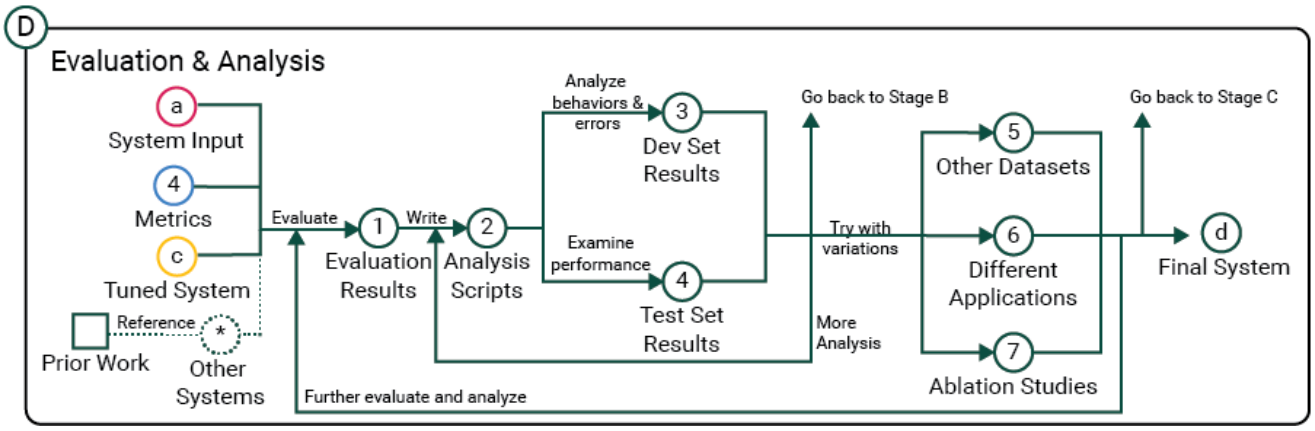

Figure 5.5: Stage D, *Evaluation and Analysis*, is the last stage of the development workflow. The system in this stage is evaluated with the full dataset, and the results are used to analyze system behaviors and errors on the dev set, as well as to examine the performance on the test set. If the system has satisfactory performance, AI developers may evaluate the system with variations.

Once AI developers have tuned the system to a satisfactory level, they evaluate the system performance with the whole **(a) system input**. In this stage, AI developers train the **(c) tuned system** with the full **(A4) training set**, and then use the **(B4) metrics** to examine the **(D1) evaluation results**. If they have other systems available (either from earlier development iterations or from other people's work, including state-of-the-art systems), they also evaluate these systems with the training set and examine the results. Note that the results are often recorded in spreadsheets as they are easy to share:

> *"Most people keep track of these, one, because these are very easy to share. Then you can go through and put whatever ones you need to put into the paper, and it keeps everything organized." – Interviewee P10*

To analyze the evaluation results, AI developers write **(D2) analysis scripts** to extract the necessary information from logs. Since systems are tuned to perform well on the training set, the details of the training set are not very meaningful. Therefore, the analysis focuses on the dev and test sets. For the **(D3) dev set results**, the analysis scripts aim to extract information from the dev set in more detail for further analysis, as these results will be used to further adjust the system. For the **(D4) test set results**, the analysis scripts can only summarize the performance in terms of the metrics, as AI developers cannot change the system based on the test set details.

There are two primary ways to further analyze the dev set results: detailed analysis and error analysis. Detailed analysis focuses on specific system outputs and behavior. For example, AI developers working on an NN model may examine the learned weights of neurons in each layer to see which neurons react to a data point more significantly. Error analysis focuses on data points in the dev set for which the system produces incorrect answers. This kind of analysis

points AI developers to potential issues that can be addressed to enhance system quality. These analyses can be performed not only on results from a single system, but also on results from multiple systems for comparison. The goal of the comparison is to understand the strengths and weaknesses of the system, and to potentially discover a way to improve the system.

If AI developers find the performance of the system to be unsatisfactory, for instance in Stage C, they may *go back to Stage B* to change the design or even design a new system. If they feel further analysis is required to understand the system behavior, they may go back to modify the scripts or write more scripts for *more analysis*. If they think the system performs well, they further *evaluate the design with variations*. This is not only to verify that the design actually works, but also to characterize the system's behavior.

Three common variations AI developers may experiment with are (1) training and testing their systems with **(D5) other datasets**, (2) applying the system to **(D6) different applications** (e.g., applying a system built for semantic role labeling to entity extraction), and (3) performing **(D7) ablation studies** (i.e., enabling or disabling each component in the AI system to see how the change impacts performance). Sometimes, when they perform any of the above variations, since the system or dataset has changed, they may need to go back to Stage C to re-tune parameters. They also need to evaluate and analyze the results as with the original version. Note that AI developers do not try all possible variations. Instead, they consider variations they believe may yield interesting results, and focus on experimenting with them.

> *"No, we do not do an exhaustive one. When we tried some combinations, we just figured that something was not working as well as other things and we just decided to drop." – Interviewee P8*

After evaluating different variations, AI developers judge if the performance is good enough. If it is, they finalize the system and deploy it. If not, they must decide whether they want to continue with the current system design or move to a new design. This decision can be difficult when the system is giving results that are inconclusive.

> *"The worst-case scenario actually is probably what we're in this project here, where it sort of works, but not really, and it sort of works, it sort of doesn't. The results aren't conclusive either way, and so in that case, then you're constantly fighting this battle of how much more time do we put into it?" – Interviewee P10*

5.5 Challenges in AI Development Workflows

According to our study, challenges in AI development workflows can be divided into two types: challenges with building models and challenges with software development. We will discuss them separately in the following sections.

5.5.1 *Challenges with building models*

5.5.1.1 Lack of understanding

Understanding how an AI system works and how changes made to the system in Stage B will impact its behavior is not a trivial task. Minor changes in the code can result in major changes in the entire system's behavior.

> *"Sometimes, the other thing that's hard about this is that sometimes when you've changed these things, that at a pretty low level in the code, and it's non-trivial to*

push that value all the way up to something that you can specify in the configuration file or in the command line or something." – Interviewee P10

When it comes to evaluation analysis in Stage D, the complex nature is also an issue. Since many times evaluations are end-to-end, it is not intuitive for AI developers to determine which components or modules have issues. AI developers must dig into the details of the components and look at the intermediate results to identify the problems, which requires considerable effort; moreover, just having intermediate results does not guarantee that AI developers will be able to understand the system's behavior.

This is especially an issue when there are limited tools available for debugging and in-depth analysis. Some AI systems, such as traditional ML models or other AI system designs that have existed for a long time, can use tools people have built over the years to facilitate analysis. For example, there are many visual analytics tools to analyze topic modeling results (e.g., [El-Assady et al. 2017, El-Assady et al. 2018, Lee et al. 2012]). However, for emerging designs, especially deep-learning-based AI systems, since people still have a limited understanding of the methods, few tools are available to support debugging and understanding deep learning models. This makes debugging harder.

"Debugging is slightly harder because earlier, we had these nice support graphs (a visualization tool they built before) that I could look at and then accordingly try to say, "Well, why is this making this mistake?" Now it's all within this deep learning library, so all I can see is it made a mistake. There is some attention (a type of deep learning modules) stuff that you can visualize. The model was looking at these

parts of the model and not looking at these parts, but debugging gets a lot more trickier." – Interviewee P2

5.5.1.2 Lack of a clear next direction

Lack of a clear next direction is another key challenge of AI development. AI developers usually make changes to their systems based on their intuitions informed by observed experimental results. These intuitions are mostly based on their experience working in the field, as well as the results of other people's work.

"But looking at the TensorBoard I can see the loss is going down. The trend, how the loss is going down. By looking at this then we may decide, okay, it was running maybe too high at the beginning so we can reduce the learning rate or the weight decay may be too small so by looking at the output graph of the loss, PR curve, and they decide which one should be more declined. And that's based on our scientific intuition." – Interviewee P3

However, when intuition does not work, it becomes very difficult for AI developers to know what to do next. They must figure out the reason something is not working, and they have to produce a remedy or improvement.

"The hardest part is when intuition doesn't work. It's like at dead end. It should be work but no." – Interviewee P7

"Figuring out what to do next is the hardest thing [in my workflows]. At each step figuring out, it was just like, I don't know what's wrong. I don't know where to look. So it was sitting around and thinking about where it might be. You look at the

results for the previous one and you just have to sit and think of, what are the ways in which ... What could have happened that would explain this, which causes another branch in the tree of possibilities. And then you go, okay, among this branch, which branch is the most likely to have happened." – Interviewee P5

This is a big issue because even when the other issues are resolved, for instance if there are good tools designed to help AI developers organize the experiments and automatically extract information from logs, they may still have no idea what they can do next to improve their systems. What they can do is keep trying and hope to discover new insights for actions. It becomes another challenging decision for them to make about whether to continue trying or completely start from scratch.

"The worst-case scenario actually is probably that we're in this project here, where it sort of works, but not really, and it sort of works, it sort of doesn't. The results aren't conclusive either way, and so in that case, then you're constantly fighting this battle of how much more time do we put into it?" – Interview P10

5.5.1.3 Lack of control

Another key issue in AI development is that AI developers do not have direct control of the behavior of AI systems. This is especially an issue for ML-based AI systems, as ML model behavior is dependent on training data, learning algorithms, and parameters. Although AI developers can define the structure and components of a model, they cannot force a model to learn a specific behavior in a fully-controlled manner.

"With machine learning, the problem ... it does tell you, 'Oh, okay, it's looking at these parts. I should change something there,' but I cannot control the behavior of what the learning system is going to learn, right? Earlier, since I was the one who decided the parameters, I can control it. Now all I can see is, okay, given the model structure, this is the parameters it's learning. I'm not happy. I can't force it to learn something else. The only thing I can do is, okay, if I change my model in this way, maybe it behaves the right way. If I change my model in this way, maybe it will ... I don't have that much control." – Interviewee P2

This is also the case for non-ML AI systems as they usually lack system-level control. For example, AI systems based on rules can also become difficult to manage when the number of rules grows too large and the rules interact with each other. Even though AI developers can determine the exact behavior of one component, the overall behavior may still not be controllable. As with the example we mentioned in Section 5.4.3.1 about the cons of pipeline systems, the impact of changes can be difficult to foresee.

5.5.1.4 Nondeterminism in results

One other challenge in building AI models is that AI systems have nondeterministic results. Sometimes AI developers make changes to the system and find the performance improves. However, they later find the improvement was not due to the changes but merely to chance, as many algorithms leverage probabilistic methods. This is not something easy to discern; verifying the results takes effort. Sometimes AI developers initially believe the changes help, and only later discover that they were wasting time pursuing this direction.

“So [it was] 0.72 but it degraded down to 0.6 something. That's terrible. The new one is, oh, precision is back up again and then recall doubled for some reason. I don't know why. I think just got lucky.” – Interviewee P5

Nondeterministic results can also arise from the use of GPUs. Certain numerical operations on GPUs can result in nondeterministic outcomes, and solutions often require an esoteric understanding of the computational techniques underlying libraries like PyTorch or Tensorflow [27]. Thus, unless the library in use specifically handles a certain operation, AI developers may find that setting a random seed does not guarantee deterministic results as is often expected when using CPUs.

5.5.1.5 Limited resources necessitate intuition-guiding design and parameter search

AI developers have to deal with a number of time and resource limitations. For example, theoretically, it would be best to attempt all possible parameter combinations and values in parameter engineering. However, since the parameter space is typically large, an exhaustive search is not a viable development approach given the amount of time and computing resources AI developers usually have. Sometimes, the number of experiments developers can conduct is limited by deadlines. If model training cannot finish before the deadline, they must terminate.

“It is kind of a greedy approach to optimize parameters. If you want to do entire grid search, then those would be million experiments and it will never finish” – Interviewee P4

“The bigger ones can go across days. So, some bigger models take so much time that you cannot run hyper optimization on it.” – Interviewee P6

AI developers must decide how to utilize their limited time and resources, which limits the number of directions they can explore in system design. As a result, they must rely on intuition to guide their design and parameter search.

5.5.2 *Challenges with software development*

5.5.2.1 Difficulty in experiment organization

One key challenge of AI development is the difficulty in organizing experiments, which slows down the experiment and analysis process. AI developers do not know ahead of time what experiments they will need to conduct. Instead, they usually conduct a few experiments first, and then decide what directions they want to try next:

> *"Because I don't know what all experiments, I need to try in one go, right? I'll try some experiments, then try some more, try some more." – Interviewee P2*

Even if AI developers try to use a pre-defined experiment organization (e.g., group experiments by different datasets), the direction of their experiment may change so much at a later point that the original way of organizing experiments no longer applies (e.g., they decide to use only one dataset and thus grouping by datasets is not helpful anymore).

Tools for setting up or analyzing experiments thus must be flexible when structuring experiment-related information. Adding such flexibility is not an easy task. For instance, experiments can be organized based on any attribute, such as by datasets, by the presence or absence of a component, and by the size of the training set. Making a backend application programming interface (API) that can take different attributes for organization is not a trivial task, and may not always be feasible.

5.5.2.2 No generalizable log format and contents

Another challenge in AI development is the lack of a generalizable log format and contents. Since AI developers' queries to the logs may change depending on their analysis questions, the output format and contents can be different case by case:

> *"I think the more challenging part is running all of this (the experiments) and saving all of the logs in a structured form, and also maybe analyzing all of these logs together. Even if I'm able to group them, I need to be able to see all of the metrics that I care about, across all of the experiments in some standard format, to be able to compare results. Also be able to run them automatically." – Interviewee P6*

Since there is no generalizable format for the logs, AI developers may not have the tools to automatically record and summarize all evaluation results for analysis. Instead, they often use spreadsheets to record their experiment results manually, but this is an error-prone process. Even when they write scripts to automatically generate the summary, they may need to write many different scripts in order to accommodate possible queries, or they have to update the scripts every time the format changes. To AI developers, modifying scripts or writing new scripts is not challenging, but it does take time:

> *"I wouldn't say that things are challenging, but sometimes writing on these scripts to get specific aspects of these intermediate results that I'm interested in, because those are ad-hoc scripts. Those need to be written for the specific kind of information that I need to get. That usually takes some time. It's not challenging,*

it's just that depending on what I'm trying to investigate it needs some time that I need to spend on it to get results." – Interviewee P8

When AI developers initially write the (D2) analysis scripts, they may not expect that these scripts will be re-used much, and thus the design of the script may not be easy to automate (e.g., the script does not accept command-line parameters). But later, when they seek to explore further, they may have to modify the scripts for automation, which requires extra effort.

"If some parameters turn off, turn on, and so, as I experiment, I realize that oh, something I should try is turning this bit off. Then I have to go in here and I can modify my script so that it does that. And then, I modify this script to take that argument, and so it's more of like a ... I have to change the script code in multiple places each time I think of a new experiment to run." – Interviewee P5.

In addition to automation, deciding what to log or what to extract from the logs is not an easy task. Due to the exploratory nature of AI development, it is difficult to foresee future needs. For instance, during Stage B, AI developers may not know which information their systems should save for later. As the development proceeds, especially during Stage D, based on the analytical needs, AI developers may gradually cause their systems to output more information to the logs. Therefore, they need to go back to Stage B to change the system. To accommodate their evolving needs, they must iteratively modify the scripts, or write more scripts.

However, the more information is logged, the more expensive an experiment becomes in terms of time and cost. This is because outputting more information is a disk operation and thus

slows down the model and requires more space to save the logs. This trade-off is another difficult decision that AI developers must make.

5.5.2.3 Extensive manual work

An issue that comes from the exploratory nature of AI development is that many steps in the development process are done manually, which is time-consuming and unreliable. For example, when AI developers collect experimental results in spreadsheets manually in Stages C and D, and mistakes are made.

> *"To come up with this results spreadsheet, I had to look at least these many log files to look at the results and put that in the spreadsheet. If I copy and paste it in the wrong column, it's a problem… So, I don't like this. Unless I write a script that goes through all of the log files, and gets me all of the information that I format, there's no other way to do it." – Interviewee P6*

AI developers also feel this manual process is painful. Some may initially have the spreadsheet cells conditionally formatted with colors, but later find it takes too much effort so they stop doing it. This leaves inconsistency in their manual work, which can be confusing to them or other people later.

> *"Because before I had it like this to do this comparison and then afterwards when I had the new version, I just added a new column, and I realize that now the way you ... The versions are here to here to here, and I was like, 'Oh, I'm so lazy to change it though because nobody's going to read this except me.'" – Interviewee P5*

In addition, in Stage C, configuring and running batch experiments with different sets of parameter values also requires extensive manual work. For some models, due to the way their systems are designed, AI developers must manually change the parameter values in configuration files. Even given the ability to conduct batch-like grid search for parameter tuning by setting the ranges, when it comes to collecting the results, they must still do so manually to put everything in a collective place. All this manual work is time-consuming, and can lead to unintentional errors.

5.5.2.4 No visualization framework to accommodate the rapid changes

Visualization is often used to facilitate debugging and analysis tasks. However, what AI developers seek to visualize is exploratory and may not be generalizable between different models or even slightly different system designs. In addition, during Stage B, models are still changing rapidly, and visualization must change accordingly. Therefore, to satisfy their needs, a tool must accommodate rapid change.

> *"Even in this demo for example (pointed to a visualization for semantic role labeling), if you look at semantic role labeling ... like the kind of visualization I care about are very different here. Since I have just one sentence there's no new notion of alignments. This is very specific to the model and I doubt there's anything generalizable here. You just have to sit down, write some code, make sure you are visualizing things. I can't think of a very general framework where you could just change some kind of config and it will give you the visualization you care about." – Interviewee P13*

Typically, when AI developers need visualization, they must create it by themselves, which can consume a lot of time and effort, especially when the target visualization is not intuitive. Even though visualizations can help debug a system's inner workings, AI developers may not want to spend time on it. Instead, they may prefer to spend their limited time modifying the model or trying more variations. The trade-off between spending time building a visualization and spending time on iterating the model poses challenges for AI developers.

> *"I won't create a visualization of a four-dimensional matrix because, A, I don't know how to create it and, B, it's not going to be useful. A, there is some effort involved in creating these visualizations, and so you kind of have to choose. Given my limited time, what should I visualize? Then you say, 'Okay, this makes the most sense to visualize,' so you look at that. Then it becomes, okay, now what's more time ... What's the better use of my time, creating a visualization of this other layer or trying out this new thing which I think will impact the layer that I'm visualizing, right? Do you just go deep into debugging and try to visualize each and every layer, with the potential that your model might completely change? Most of these visualizations are useless. Take a visualization that's most useful to look at, and then use that to try out random things. Do more experimentation and less debugging or do more debugging and less experimentation." – Interviewee P2*

5.5.2.5 Difficulties in debugging complex code

Difficulties in understanding can also arise from the fact that AI system code can have layers of layers, and sometimes when functionalities are abstracted out as an interface, many details are hidden. Thus, knowing what is going on behind the interface can be problematic. For

instance, when the data are handled via data provider modules, the details are hidden behind the data provider interfaces. If there is any bug in the data provider code, it is difficult to detect, especially when the data provider itself is very complex. AI developers may discover these problems in a late stage of development (e.g., when trying variations in the evaluation and analysis stage) and have to trace back all the way to the data processing stage to find the problem, and in the end have to re-run all the experiments.

> *"The shuffling code is kind of convoluted, so we shuffled three times. So we shuffled twice but the third one doesn't shuffle so we found this thing and just shuffled it again and run all the experiment on this first data set on the second data set." – Interviewee P3*

Additionally, AI developers may not know what their dependency libraries actually do in detail. When the changes they want to make are related to what is wrapped in the libraries, they must dig into the dependencies to figure out the details, which can be a tedious task.

> *"We actually had a conversation about changing the initialization for one of the models, and someone had to dig through the PyTorch code, where the question was, "How does PyTorch do this?" and our answers were, "I don't really know," and so then [an AI developer] had to dig through the PyTorch source code to figure out how they actually did the initialization." – Interviewee P10*

5.5.2.6 Trade-offs between writing new code or using existing library

There is a trade-off in choosing to use existing libraries or to write new code. Using existing libraries saves time writing and testing code, but as these libraries are written by other people, it

may not be obvious how they actually work, and thus when there are issues (e.g., the initialization methods do not apply to certain datasets), it is harder to debug. Given the limited time and resources AI developers have, this is always a difficult choice.

> *"You're constantly doing this. You're constantly playing this game where you would really much prefer to be writing your code and thinking about problems on a higher level, but in order to get things actually work in practice, you more often than not have to go down to a very low level, and that can involve even going all the way down to the low levels of the open source libraries that you're relying on." – Interviewee P10*

5.5.2.7 Difficulties in tracking changes in code and configurations

Since the training steps in Stages B, C, and D all take time to run, by the time they finish, AI developers may not remember what they changed or which parameters they used.

> *"So I may have sometimes when you run an experiment, and then you run another experiment and then you forgot what exactly I changed." – Interviewee P11*

> *"Yeah, yeah so right now I might remember okay, what are those other parameters and what I've used. But probably two months later I won't remember those. If I want to do exactly the same experiments, it's important to give those as well." – Interviewee P12*

In the case of initial development (Stage B), AI developers may iteratively change the system based on findings from the sanity check but may forget what they have changed. In the case of parameter tuning (Stage C), they may try a range of different values for each parameter,

but forget which experiment corresponds to which parameter. During evaluation and analysis (Stage D), AI developers may even try new datasets or want to understand the effect of a single component. The large number of configuration combinations makes it difficult to remember everything. Some AI developers use Git to keep track of versions, but some files may not be checked into their repositories (for instance, developers may not save every version of their datasets due to the large file sizes).

Some AI developers use directory names to note the changes, but when there are too many changes, it is impossible to write everything in the limited space for text. Hence, they must still examine the configuration files, the code and the dataset to figure out what changes were made, which again can be time-consuming.

> *"It's mostly through naming variables, naming directories, and making sure that my config files have enough information." – Interviewee P13*

For AI developers who use cloud computing resources available at the research site, they must wrap their system as a job for submission to the site's job scheduling system. When creating a job, AI developers can specify a description of the job to note what the job is about. As with the directory name, some AI developers use the description field to record what changes or configurations they have for the corresponding model. Again, though, this field has limited space for text, and it is impossible to note everything. Therefore, some details may be missing from the description. If AI developers revisit the job after a while, they still need to spend extra time to figure out what is included in the job.

5.5.2.8 Issues in adapting to dependency changes

Dependencies are another challenge in AI development workflows. AI development relies not only on existing libraries but also on servers that run AI systems. During development, AI developers can find that a system that was working yesterday breaks today because dependencies in the system were upgraded and the new version is not compatible with the original version.

> *"Oh yeah, and in some cases, you'll have very disturbing things where you're like, 'My code runs great,' and then it doesn't. Like, 'What happened?' [Another AI developer] ran into these problems. Everyone has. I run into these problems, where it's like, 'Why? I'm running the same thing that I ran last week, and I can't reproduce the results. What happened?' And it turns out at some point, it's an environmental thing, or you upgraded your [dependencies]..." – Interviewee P10*

Dependencies are another reason for not being able to reuse designs or systems from other people's work in Stage B: To use other people's code, AI developers must find a way to inject and accommodate their dependencies, but it can take time to set up the dependencies and can lead to conflicts. Sometimes, using other people's code is not even feasible, as not all packages can be run across platforms. Even when AI developers want to set up other people's models for comparison in Stage D, it can be challenging and thus analysis becomes limited.

> *"The part that I feel like is really missing from my life is an easy way for people to transfer their modules, share, and not just for publication purposes. ...I want a single point of contact where I can go run the command line and get the results and not fail because I'm Python 2, Python 3, or because we don't have access to the*

data or because the username and password don't match, these things can all hurt and break a lot of pre-scripted things." – Interviewee P11

"The biggest thing for me is the ability to start and stop different components of a thing (running the ablation study) without worrying about dependencies." – Interviewee P5

Since AI systems usually take time to run, they are susceptible to random errors that may occur in the server environments. For example, a system that takes a few days to run will crash if the server goes down for updating or rebooting. Sometimes dependencies may have bugs that are not discoverable in small sets of data, but which can break the system when run with the full dataset.

"That's what the experiment really is. It's like a program that someone has made, and then it runs and the failed status usually means something went wrong in the execution of that program, whether there was a bug in the program causing it to crash or like something in Kubernetes went haywire and the thing just disappeared on us. For some reason this thing was running and then something went wrong, and now it's not running." – Interviewee P1

Sometimes, these crashes are difficult to foresee. For instance, one interviewee mentioned that his computer ran out of memory once, due to a very specific version of the GPU driver.

"It was running out of memory was because there is a bug in the very specific version of the Nvidia driver that he was using" – Interviewee P10

In other cases, there may be issues with GPUs and job scheduling, and it is difficult to predict when a job will finish (if ever). Thus, some AI developers submit duplicate jobs to increase the chance of completing their experiment.

> *"It's like you try to run say three instances of the same model. You just wish one of them will finish give you results that's it." – Interviewee P9*

5.6 Discussion

Based on our findings, AI developers still lack support in terms of tools to facilitate experiments and analyses. Many issues are caused by the exploratory nature and complexity of AI development. This leads to many opportunities for HCI researchers and designers to propose new designs for tools to assist AI development. To make them useful, it is also important to make sure the tools fit into existing workflows. Therefore, our proposed framework can help designers examine and justify how to position their designs. In this section, we highlight a few design implications based on our findings and our proposed framework for designers to consider.

5.6.1 *Flexible experiment organization*

Since AI development is exploratory, any tool designed for this field must be flexible in terms of how experiments are organized. Theoretically, an experiment should belong to a project and can be organized into a group of related and comparable experiments. For instance, all experiments for parameter tuning an AI system should belong to the same group. However, since the direction AI developers seek to explore can evolve over time, designers should not assume an experiment can only belong to one project or one group of experiments. For example, an AI developer may run three groups of experiments for tuning parameters for three different AI

system designs, and want to find the best one of each set. If an analysis tool is designed to assume comparison can only be conducted within a group, then it cannot meet this AI developer's need.

Some flexible experiment organization methods designers could consider include allowing one experiment to belong to many groups or projects, supporting reorganization and duplication of experiments, enabling tagging and grouping by tags, and ad-hoc grouping.

5.6.2 *Standardized log formats*

To avoid time costs and potential errors while writing manual logs during AI development, automation of log extraction and analysis is critical. Currently there are no standardized log formats, but given a good tool to summarize results or to facilitate analysis, AI developers could modify their log outputs to fit the tool, or use a converter for log formats. For example, TensorBoard was originally designed for TensorFlow logs, but its usefulness has led to its adoption among users of other libraries, like PyTorch. Additionally, an open source library, *TensorboardX* [9], has been created to convert other ML frameworks' logs into TensorBoard-accepted format. This demonstrates creation of useful tools can foster standardized log formats.

In addition, there are several benefits to having a standardized log format: First, it can reduce the need to invent project-specific logs for common tasks such as tracking metrics during training. Second, it can reduce confusion when revisiting logs from past projects. Third, it increases reusability of analysis scripts that expect the same log format. Last but not least, it

[9] Tensorboard X: https://github.com/lanpa/tensorboardX

increases the potential for others to develop visualization or analysis tools that expect a standard log format.

5.6.3 *Extendable visualization suite*

Prior work has proposed many visualization tools for a variety of different AI development tasks, but most of these are research prototypes which are not generalizable and reusable. Usually when building research prototypes, there are some example datasets and models the prototypes use to validate the design ideas, but the ways data or models are served can be very specific to the research projects. Even when the visualization tools are open-sourced, reusing them requires considerable effort. Therefore, there are not yet many visualization tools from research that get used in real AI development workflows.

We view the above issue as a gap between visualization research and real-world usage; our proposed framework may be a good starting point for bridging the gap. The framework can be used to identify which tasks a visualization tool should support and which key elements to include. In addition, the framework can be used to ensure such a tool is positioned correctly and streamlined into existing development workflows. For example, a visualization tool designed for the initial development stage should focus more on model internal details for debugging, whereas in the evaluation and analysis stage, it is more useful to visualize model behavior on the dev set to help find patterns. It is our hope that this framework can accelerate the modularization of these visualization tools as components for reuse, leading to an extendable visualization suite for AI development.

5.6.4 *Version control*

Since AI developers typically continue to make changes, it is important to help them apply version control to the changes, especially since these details can be easy to forget. This includes tracking all relevant information, saving info in a meaningful and organized manner, and enabling the ability to revisit info at a later time. Example tools such as Hindsight [Patel 2012] can be a useful starting point, but Hindsight applies only to code changes; AI developers need help keeping records for all the parameters, configurations, and data versions.

In addition, the changes should be saved in an organized way and should be easy to retrieve. This will help AI developers to recall the goal of their changes without the need to manually record every detailed change. For projects that use Git for version control, AI developers can use "git diff" to see what changes they have made. However, more effort can be put into providing a summary or a dashboard interface to show an overview of changes, rather than just a set of differences for each individual single file. For example, for two different configurations, a summary of the differences can tell AI developers how many parameters are different between the two, and what they are, instead of requiring AI developers to see changes at a character level.

5.6.5 *Better understanding of the socio-technical ecosystem aspects of AI development*

Another implication we want to highlight is the need to better understand the socio-technical aspects of AI development. For many years, the field of computer-supported cooperative work (CSCW) has been studying so-called cyberinfrastructure [Atkins 2003] or socio-technical systems [Baxter and Sommerville 2011], which focus on how humans interact and collaborate while using computing technology. Much of this research has been conducted at research sites

with a computing center, such as supercomputer (or high-performance computing, HPC) centers [Chen et al. 2016, Chen et al. 2019, Karasti et al. 2010, Lee et al. 2006].

Even though many AI models run on cloud computing facilities, which are different from supercomputer centers, they are both complex computing facilities, and the software and human socio-technical systems are also complex. In previous research on how scientists use supercomputers [Chen et al. 2016], a number of issues identified were similar to those AI developers encounter today. For instance, as the complexity of supercomputers increases, debugging becomes more difficult, which leads to unexpected failures and failed computing jobs. This is very similar to job failures in the AI development workflow; both lead to users (scientists/AI developers) submitting multiple jobs (and potentially overburdening the system) just in case jobs fail. Likewise, both scientists using supercomputers and AI developers are affected by dependency changes.

These findings from our study align with Forsythe's argument that the construction of AI systems is a social process. Designing tools for AI developers is much more than a technical issue; instead, designers must consider the specific ways humans interact with their technology and build tools that enable more effective utilization of both limited human time and limited computing resources. More research that studies how to better support AI development from a socio-technical system viewpoint would be extremely valuable.

5.7 Limitations

As this framework is drawn from a single AI research institute, there are obvious limitations in scope. Thus, we view our framework as an initial effort to describe AI development

workflows. In this section, we highlight three aspects of current limitations that can be further explored for extending the framework.

5.7.1 *Data quality*

The interviewees in our study usually work on datasets that have higher quality than other domains, such as medical domains. Therefore, in our framework, Stage A is relatively simple and the transition between other stages to Stage A is not emphasized. However, given our understanding of domains with noisy datasets, AI developers may spend a lot of time iteratively discovering issues in their datasets. Thus, how to include data quality as a factor in the framework will be an interesting direction for further research.

5.7.2 *Supervised vs. unsupervised learning*

Although we do not intend to focus on supervised learning, most of our interviewees were working on building AI systems with datasets with ground truth labels. Some may use unsupervised learning during Stage B to explore datasets or the problem but did not go into detail about how their workflows for unsupervised learning differed from that for supervised learning. Thus, our framework may be more applicable to describe supervised learning workflows than unsupervised learning workflows. Elements such as splitting the “training/dev/test” sets do not exist in an unsupervised learning context, whereas key tasks such as parameter tuning are still meaningful in unsupervised learning (e.g., clustering problems require determining the number of clusters). We expect future work to deepen the discussion of how using unsupervised learning changes AI developers’ workflows.

5.7.3 *Research vs. industrial application contexts*

Since most of the interviewees in the study site work on AI systems for research, the framework may be limited to AI systems that are under research. However, since the boundary between AI research and AI applications is vague (many AI systems in industry come from industrial research labs such as IBM Watson), we expect to see shareable workflows between the two contexts. On the other hand, the difference between the two contexts may become more significant after AI development matures. Therefore, we suggest conducting more studies on AI development workflows in the context of industrial applications to further verify and modify the framework to ensure the workflows described in the framework are up to date, and clearly differentiate between the two contexts.

5.8 Conclusion

In this chapter, we propose a framework to describe AI development workflows based on a series of on-site contextual interviews with AI developers. We highlight challenges that AI developers encounter and provide a set of implications for designers and future research. For example, we point out the need to support flexible experiment organization for any experiment-related tool design. We also recommend that future research should involve a better understanding of the socio-technical ecosystem aspects of AI development.

The results from this study can help designers to understand the dynamics of AI development workflows and better position new tool design in context. We also offer a number of directions for future research. In addition, since we view this work as an initial effort to provide a high-level overview to call for more attention from designers and researchers to design

tools for AI developers, we want to encourage further extensions or modifications of this framework to include more diverse use cases.

Although this chapter does not directly focus on visual analytics, the results from the studies can inform VA design in the future. It can also be used to analyze existing VA tools for AI development to identify gaps and opportunities for further research. For example, QSAnglyzer was created for Stage D in the workflows; to make QSAnglyzer more useful, it could be connected with functionality that allows making changes to the QA solvers to streamline the workflows. This is similar to AnchorViz, which can be used in Stage B for initial development or in Stage A to further understand the model and data, making iML not only a new paradigm but also an approach to enhance the quality of traditional ML development workflows.

The rapid growth in AI development underlines the importance of understanding how people develop AI systems as well as the importance of building tools to support this process. With the framework and the identified challenges, we hope to drive better support for AI development, which can free up human time to improve the efficiency, diversity, and ethics of AI systems.

Chapter 6. Conclusions and Future Work

In this thesis, we tackle three different aspects of AI development workflows: evaluation analysis, interactive machine learning, and general development workflows. We use a human-centered design approach to study AI developers and come up with the designs of both QSAnglyzer and AnchorViz. The interview study on current AI development workflow summarizes the current practice and highlights challenges of existing workflows, suggesting research opportunities for future designers. The framework for AI development workflows can be used to clarify the design contexts and goals of newly proposed tools.

The results from this thesis support our thesis statement: In QSAnglyzer and AnchorViz, we demonstrate how VA can support data exploration and analysis in AI development workflows. Specifically, in Chapter 3, we illustrate how the visual representations and interaction techniques of QSAnglyzer help QA developers find patterns and conduct analytical tasks faster and more accurately. In Chapter 4, the design of AnchorViz also helps AI developers to discover more feature blindness errors than automatic methods. In addition, the results from the interview study in Chapter 5 highlight many challenges AI developers encounter in current AI development practices. The framework and challenges we extract from the study can help future designers and researchers to create tools to further improve existing workflows.

In this conclusion chapter, we review key findings from my previous chapters and the corresponding research questions and highlight a few directions for future research. In addition, we use sensemaking theory to contribute to a theoretical foundation for tools to support AI development, in evaluation analysis, iML, and AI development workflows. The goal is to apply sensemaking theory to inform visual analytics design for AI development. Last but not least, we

conclude this thesis by reflecting on what we have learned along the way and provide some recommendations for future researchers and designers.

6.1 Key Findings

We investigate two types of research questions: understanding AI developers (RQ1, RQ4) and designing visualization tools for AI development (RQ2, RQ3). In Chapter 3, to understand AI developers, we study how they conduct evaluation analysis to collect design requirements. In Chapters 3 and 4, we turn our attention to the design of visualization tools and propose QSAnglyzer and AnchorViz as example visual analytics tools to facilitate AI development. Then we take a further step in Chapter 5, investigating AI development workflows in general, and highlighting issues developers encounter.

In this section, we summarize key findings on a chapter-by-chapter basis. From the study in Chapter 3, we identify a few key requirements for evaluation analysis such as where to focus the analysis, the usefulness of categorization, metrics, and the need to support complex inquiries.

1. **Input data and how models perform on these data is the key unit of analysis (in the case of question answering, the input data is questions).** In other words, AI developers try to slice and dice the input data to look for patterns during their analysis. This aligns with findings in Chapter 5: in the design evaluation and analysis stage, AI developers seek to analyze model behavior and errors. Model behavior includes both how the model performs on a given dataset as well as how behavior changes when the input changes. Error analysis focuses on examining data points for which the model predicts incorrect results. The two analyses both center on input

data and their relations to model performance.

2. **Categorization helps AI developers slice and dice evaluation results to further understand model behavior patterns.** Since aggregating metrics provides only a summary, and single point details do not cover a big picture, AI developers need an in-between method to examine evaluation results.

3. **Metrics are useful for assessing evaluation results not only at the overall level but also at the group level.** In the case of QA systems, accuracy is the key metric, and AI developers examine its relationship with other metrics such as the number of questions in a group and solvers' accuracy variance within a group. These metrics are useful for assessing not only the overall performance but also the group-level performance.

4. **AI developers form complex queries on evaluation results, whether it is the result of a single experiment or a comparison between multiple results. Different levels of details are required as part of the complex queries.** This also aligns with findings in Chapter 5, and explains why AI developers must keep changing their analysis scripts: their queries on the results and the logs keep evolving throughout the development process. In QSAnglyzer, we enable filtering on multiple angles by clicking as well as searching, and we place intermediate outputs inside data table details to provide flexibility for showing details regardless of the format.

In Chapter 5, the key findings include the framework of the AI development process itself as well as the challenges AI developers face, including experiment organization and the need for generalizable log formats and visualization suites, as well as change tracking.

5. **AI development workflows have four iterative stages: (A) data processing, (B) initial development, (C) parameter engineering, and (D) evaluation and analysis.** The four stages have different goals, key elements, and tasks. In Stage A, the goal is to process data into formats for AI systems to take. Key elements in this stage are datasets, which can be newly collected datasets or existing datasets. Tasks in this stage focus on processing data and splitting the dataset into training, development, and testing sets. If the training set is large, AI developers may select a training set for representative examples. In Stage B, the goal is to implement a newly designed AI system or modify an existing AI system for the target goal of the system. In this stage, the key elements are the models the developers are working on, and the tasks concern implementing and debugging the model, and making sure the models work. In Stage C, the goal is to find the right set of parameters for the models, and thus the key elements are the parameters and the tasks are parameter selection and parameter tuning. Lastly, in Stage D, developers focus on evaluating the AI system and analyzing evaluation results. The key element is the evaluation results, and the tasks are evaluation analysis as well as variations for further understanding.

6. **Tools for AI development require a flexible way to organize experiments and related information.** Since directions taken in AI development can change

significantly, it is important that tools designed for AI development, especially evaluation analysis tools, to have a flexible way of organizing experiments. Some potential design choices include enabling resignation from or multi-membership in different groups, using tags, allowing ad-hoc queries, and supporting experiment duplication.

7. **Discovering the need for shared log formats and an extendable visualization suite.** One challenge in AI development workflows is that there are no shared log formats, which makes automatic extraction and summary difficult. There is also no extendable visualization suite available to convert existing visual analytics tools for ML/AI into reusable components. However, it is possible that a useful visualization tool can promote shared log formats, and more visual analytics tools can be proposed once there are standardized log formats. Thus, the visualization suite and log formats should be considered hand-in-hand.

8. **Highlighting the need to track changes.** Since AI development is usually exploratory and models take time to run, it is easy for AI developers to forget what they are experimenting on; thus developers will benefit from change tracking tools as well as tools that clearly summarize the differences between two versions.

In addition to the findings from studying AI developers in Chapters 3 and 5, a few key design contributions made from the design of QSAnglyzer (Chapter 3) and AnchorViz (Chapter 4) are listed below. The key contributions of QSAnglyzer are the design of prismatic analysis and the QSAnglyzer interface elements. The key contributions from

AnchorViz are the idea of creating “anchors” to represent concepts and the corresponding interface design.

9. **Proposing prismatic analysis and the corresponding visual analytics design.** QSAnglyzer is a powerful tool for analyzing QA system evaluation results, as it leverages the idea of prismatic analysis and enables users to perform complex queries visually. The idea to provide multiple ways by which to categorize datasets; using the same metrics to summarize them facilitates comparison between various angles and within groups. In addition, the visual interface and the interactivity of the tool provide a tangible way for users to explore and come up with hypotheses, and directly verify within the tool. According to the user testing results, users conduct analysis tasks more accurately and more quickly. Expert users are also able to find new insights within an hour of use (Section 3.6).

10. **The use of the “overview first, zoom and filter, details on demand” mantra to present the panels in QSAnglyzer and show them on the same page.** By following this mantra and putting all three steps on the same interface (but in different panels), QSAnglyzer provides a useful overview of the performance of different models. The Angles panel is where users zoom and filter the dataset, and the Question Table panel is where details are found. Since everything is on the same page, it is easy for users to maintain context. We made the design decision to present all three of the panels on the same page, which makes it easier to maintain context.

11. **The idea of using “anchors” to slice and dice datasets.** In AnchorViz, one key

design principle is to use “anchors” to spread out the dataset on the interface so that users can explore visually. Color and shape encodings highlight inconsistencies between existing labels and prediction; users create more anchors to elaborate a concept, and use these anchors to further slice and dice the datasets to find more errors. The user study shows that AnchorViz helps users discover more prediction errors than stratified random and uncertainty sampling methods (Section 4.5).

12. **Inventing the layout algorithm of AnchorViz.** In addition to anchors, the layout algorithm of AnchorViz is also critical for error discovery. By use of a similarity-based layout algorithm (Eq 4.2) on a polar coordinate system, users see the impact of the anchors on a 2D plane. Without this algorithm, the similarity between anchors and each data point forms a high-dimensional space which can be difficult to understand.

13. **Metrics for anchor effectiveness.** The new metrics we propose—anchor error precision (AEP, Eq. 4.4) and anchor error recall (AER, Eq. 4.5)—are another contribution of AnchorViz. These metrics may be used in future research to measure the effectiveness of anchors in terms of discovery effort, and potentially lead to new automation methods to help select anchors.

In summary, this dissertation not only deepens the understanding of AI developers and their workflows, but also proposes visual analytics design to demonstrate how visualization can enhance AI developers’ experience in building AI systems. Based on the findings, we want to propose three directions for future research.

First, we suggest future research should focus on creating an extendable visualization suite for AI development. This direction is not only based on the challenges AI developers encounter that we found in Chapter 5, but on our own reflections after working on QSAnglyzer. We describe these in detail in section 6.2. Second, er suggest future work should continue the effort to make AI accessible to a broader population. This is one of the goals of interactive ML and building tools to support AI development. This is also what AnchorViz and the framework we proposed in Chapter 5 should assist in enabling. We describe the details in section 6.3. Finally, we suggest using sensemaking theory to contribute to a theoretical foundation for tools to support AI development. The theoretical framework proposed in Chapter 5 can help structure our understanding of issues AI developers encounter. Through working on the three projects, we also observed that sensemaking is a process that can be used to describe some of the AI developers' processes. In section 6.4, we use sensemaking theory to scaffold the understanding of AI evaluation analysis, interactive ML, and the whole AI development process. Our hope is to seed future research in these areas.

6.2 Direction for Future Research (I): Creating an Extendable Visualization Suite for AI Development

As we suggest in Section 5.6.3, future research should focus on creating an extendable visualization suite for AI development. Currently, AI developers often use environments such as Jupyter Notebook[10] (an interactive programming environment widely used in data science) or integrated development environments (IDEs) to implement their AI systems. Visualization often

[10] Jupyter Notebook: https://jupyter.org/

exists in environments such as an output cell in a Jupyter notebook or a static image in a directory. Researchers have proposed many visualization systems that showcase its power, but most of these are not available to AI developers since most of them are research prototypes and are not easy to fit into existing AI development environments. Creating an extendable visualization is not merely an engineering problem, but a research problem, as it requires a deep understanding of the AI development ecosystems. For instance, instead of creating a new web application, creating a widget that AI developers can use in the Jupyter Notebook environment can accelerate the adoption of such a tool (e.g., The What-If tool [Wexler et al. 2019] is available not only as a web application but also as a Jupyter notebook widget[11]).

When we worked on QSAnglyzer, the tool was designed based on the very specific setup of the AI developers we studied. Even though there are other AI developers in other research institutes who work on QA systems, QSAnglyzer cannot be directly adapted to their workflows. Therefore, we suggest that future research should focus not only on designing more visualization tools for various AI systems and workflow tasks, but also on understanding the dynamics in AI development. This is not to say we cannot completely reinvent the experience of AI development by creating a brand-new set of tools, but that designing such tools requires thorough studies to understand AI developers' needs in context.

[11] What-If tool demos: https://pair-code.github.io/what-if-tool/index.html#demos

6.3 Direction for Future Research (II): Making AI More Accessible to a Broader Population

Another direction we suggest for future research is to make AI more accessible to a broader population. This is actually the ultimate goal of improving the AI development processes and experience. For instance, there were no personal computers a few decades ago. People used punch cards to write code, but punch card readers were limited. As a result, programming itself was not accessible to even professionally educated programmers. Then, after personal computers become pervasive, programming became accessible to programmers, but still not accessible to people with a limited background in computer science. In addition, difficulties in code development such as complex syntax, no user-friendly editors, and limited learning resources still hindered many programmers. Currently, more and more programming languages and tools have lowered the barriers of programming (e.g., Scratch [Resnick et al. 2009], Alice [Dann et al. 2008]), and many graphic user interfaces (GUIs) aim to reduce the burdens of developers (e.g., many IDEs check if the use of a function matches the function' definition). Interactive environments such as Jupyter Notebook and JS Bin[12] enable developers to quickly see the results of their current code and make modifications. There are also visualization tools that help explain the behavior of computer programs (e.g., [Diehl 2007, Guo 2013]). Although programming is far from fully accessible to anyone in the world, these efforts for increasing accessibility to novices and end users can also be good references for making AI accessible to a broader population..

[12]JS Bin: https://jsbin.com

Existing efforts to lower barriers to AI development include interactive ML (iML), explainable AI (XAI), and end-user ML. Using iML, we can better confirm that ML models are learning what we want them to learn; using XAI, we can better trust AI systems, facilitate the debugging of AI systems, and ensure fairness; using end-user ML, people without a professional background can learn and build their own models for their personal use. A more detailed literature review can be found in Chapter 2. AnchorViz is only one example that demonstrates how interactive visualization can help people who do not necessarily know ML deeply to discover errors in an iML classifier. Future research should continue the above efforts to make AI systems a type of technology that everyone in the world can create and use.

6.4 Direction for Future Research (III): Using Sensemaking to Inform Visual Analytics Design for AI Development

The last direction we suggest is to connect the empirical findings with a theoretical framework—sensemaking theory. In this section, we provide an initial attempt to connect a notional model of sensemaking theory with AI development, explaining why QSAnglyzer and AnchorViz work. More studies should be done to extend and iterate the modified notional model to come up with implications for VA tools for AI development.

6.4.1 *Background—Sensemaking theory*

Sensemaking is a process by which humans come up with an understanding of a situation they face. The term originates from organizational studies [Weick et al. 2005], and specifically describes how humans figure out a plausible explanation for their experiences. The term has been extended for use in the HCI and VIS community, especially for making sense of data. One

well-known theory of sensemaking is that of Pirolli and Card [Pirolli and Card 2005]. According to Pirolli and Card, "*sensemaking tasks consist of information gathering, re-representation of the information in a schema that aids analysis, the development of insight through the manipulation of this representation, and the creation of some knowledge product or direct action based on the insight.*" Figure 6.1 shows their notional model, which they devised based on their study of analysts. The model basically has two sub-loops: a foraging loop for data collection, and a sensemaking loop for coming up with a representation of the data. The loops are iterative, and all the steps can go back and forth. This theory is one of the foundations of the visual analytics field that explains why visualization can be helpful when carrying out analytical tasks.

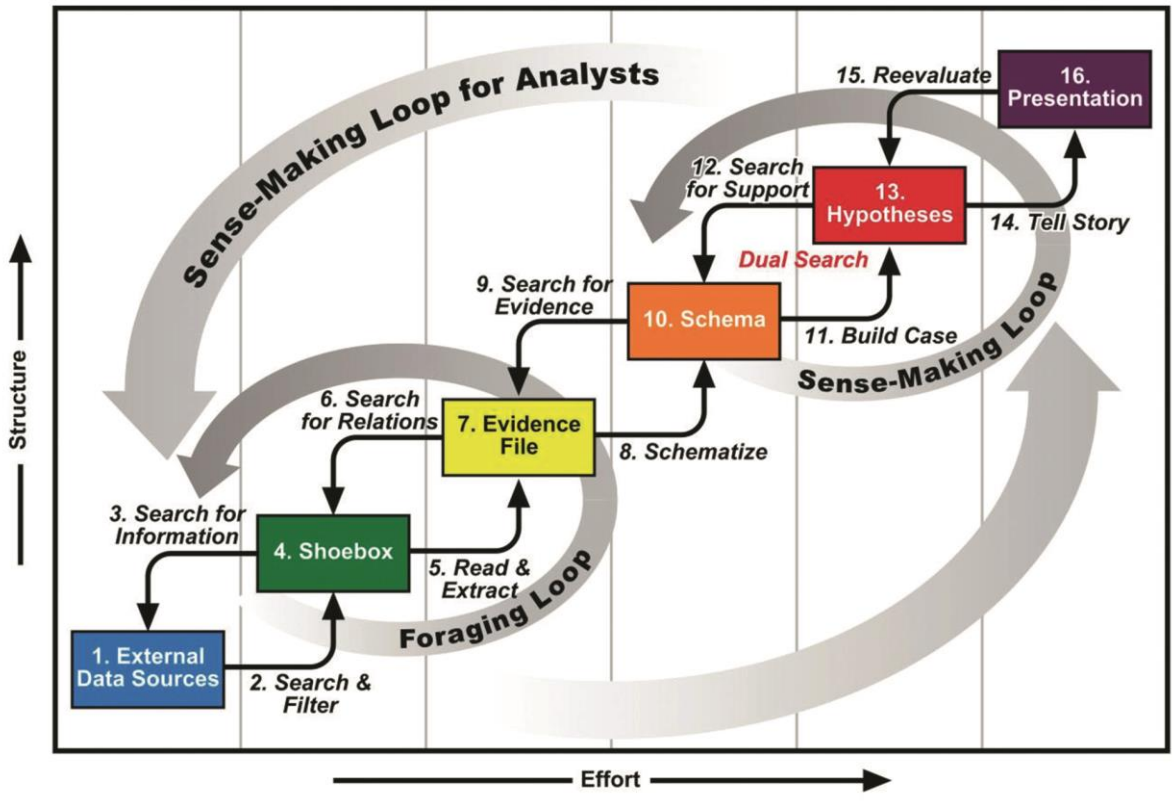

Figure 6.1: Pirolli and Card's notional model of sensemaking [Pirolli and Card 2005]

Another well-known sensemaking theory is the data-frame theory by Klein et al. [Klein et al. 2007], which is illustrated in Figure 6.2. Unlike Pirolli and Card's model, which focuses on the process, the data frame theory puts the "frame" at the center. Frame is similar to schema in Pirolli and Card's theory, and all the tasks revolve around the frame. Since we am more interested in the perspective of process, we will use Pirolli and Card's theory as our primary theoretical foundation.

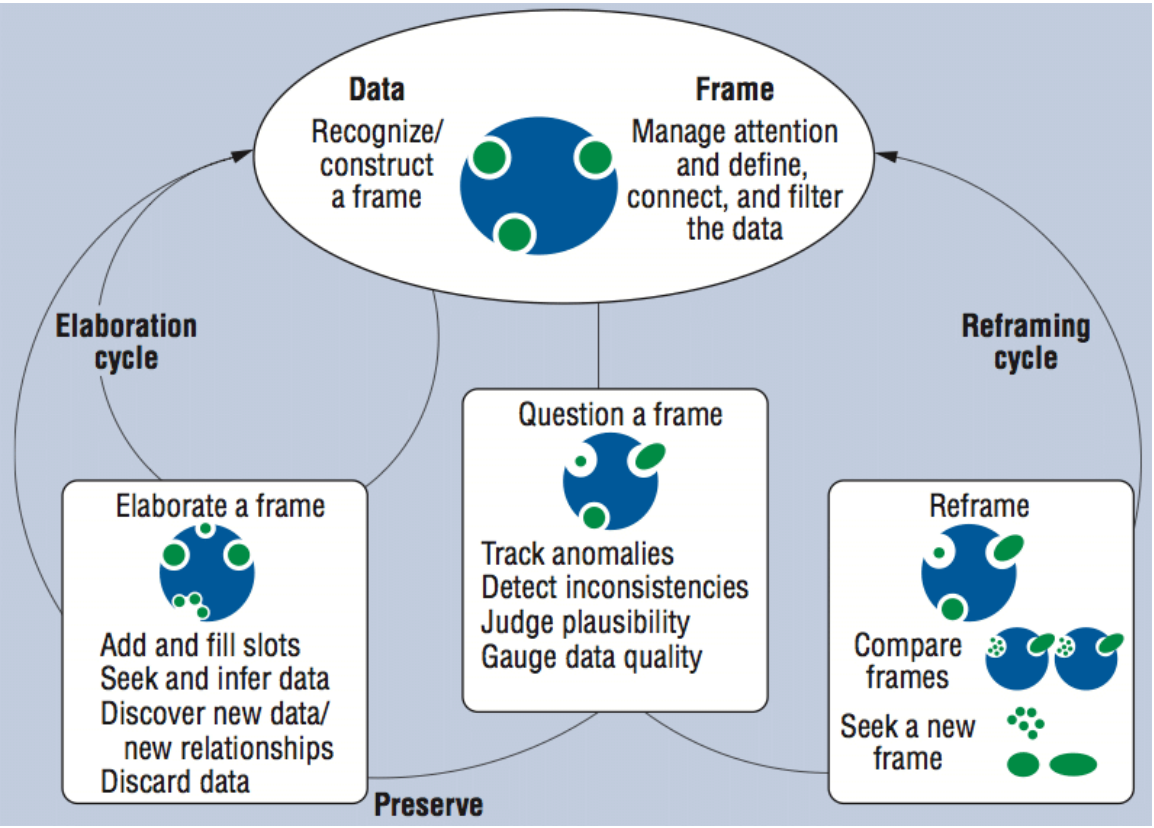

Figure 6.2: Data-frame theory of sensemaking

6.4.2 *Sensemaking in evaluation analysis*

In evaluation analysis, sensemaking is an important task to gain insight about a system's behavior, especially the pros and cons of each design. Based on the user studies in Section 3.3,

we modify Pirolli and Card's model to describe the sensemaking process in evaluation analysis (Figure 6.3).

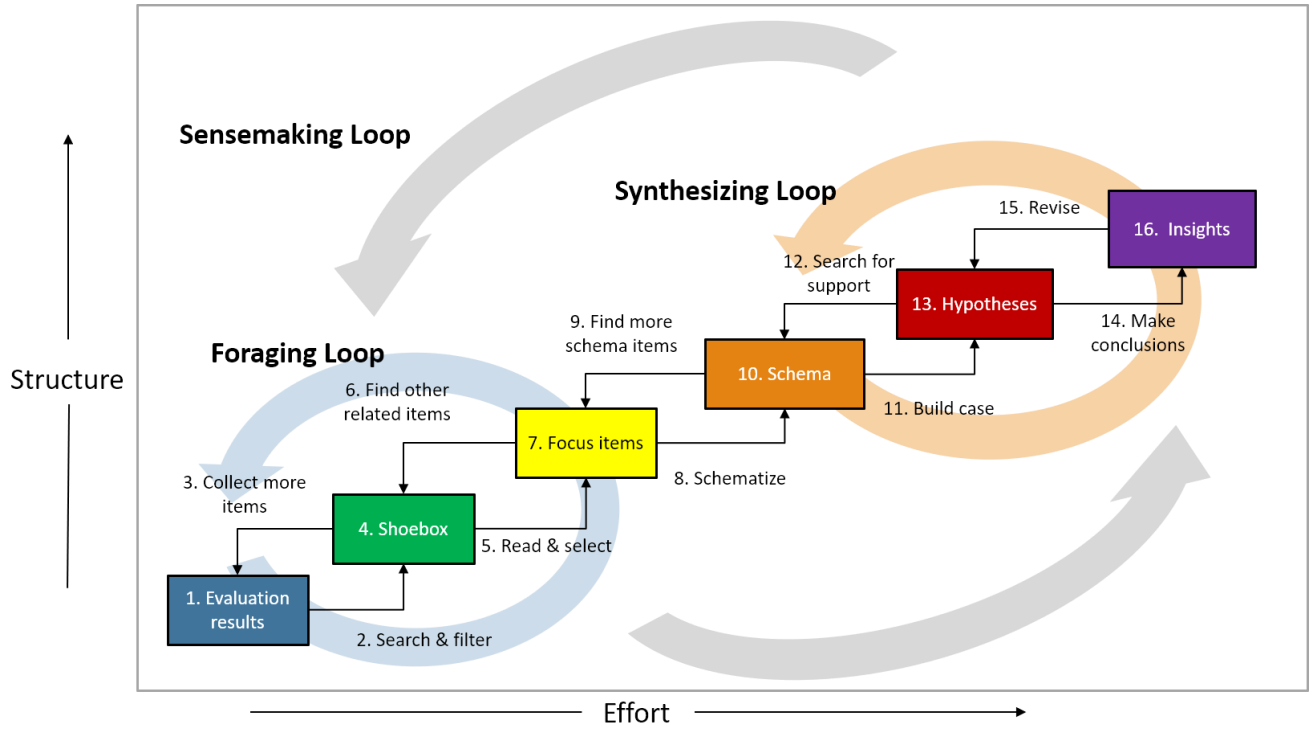

Figure 6.3: Modified notional model for sensemaking in evaluation analysis

In the modified version, unlike the original version for which the goal is to make sense of the data sources and come up with explanations for a case, the goal of the sensemaking process in evaluation analysis is to make sense of the evaluation results and generate insight to determine the qualifications of the system and/or possible design modifications. Therefore, the key data sources are **(1) evaluation results**, which can range from evaluation metrics and intermediate model outputs to any other data related to evaluation. Once AI developers have these data sources, they **(2) search and filter** to get data that are potentially useful. For example, AI developers can have as many metrics as they want, but they focus only on those few that matter to their development goals, which are collected into the **(4) shoebox**. Then they **(5) read and**

select items from the shoebox to identify items that they want to focus on examining. These **(7) focus items** are the basic units from which they glean insight. AI developers also **(6) find other related items**, that is, items related to the focus items in the shoebox to collect more examples. If they cannot find enough items in the current shoebox, they may also go back to the evaluation results to **(3) collect more items**. The above steps form the so-called ***foraging loop***.

Once AI developers collect the focus items, they attempt to **(8) schematize** these items. In other words, they attempt to determine groups or categories that qualify the meaning of these items, to compile the so-called **(10) schema**. For example, in the case of QA systems, they may find a group of items that represents the topic of questions that their QA system cannot handle. With the schema, they **(11) build cases** to come up with **(13) hypotheses** about why their systems are not working well. Given these hypotheses, they **(12) search for support** from the schema, and if the existing schema does not cover what they are looking for, they return to the foraging loop to get more supporting items. AI developers can iteratively verify and modify the hypotheses. If they think their hypotheses are correct, they **(14) make conclusions** to come up with **(16) insights**. If they find these insights are not useful or accurate, they go back to further **(15) revise** them. These steps form the ***synthesizing loop*** of the sensemaking process.

Many design elements in QSAnglyzer support the above sensemaking process, which explains why QSAnglyzer works well in QA evaluation analysis. First, each angle is a way to categorize questions, and categorization is one way to schematize items. Therefore, by showing and enabling interaction with these categories, AI developers can speed up the process of creating a schema. This can help AI developers build cases for hypotheses more easily. Second, with the representation of schemas, AI developers can tangibly use schemas to slice and dice the

evaluation results, which also supports the foraging loop. For example, AI developers can click on a certain category to filter to questions in the category and examine the results within that category. This helps collect interesting examples (AI developers can use the “star” function to collect examples), which can lead to creation of new schemas. Furthermore, simultaneously showing multiple angles and multiple categories helps AI developers to compare and contrast these schemas more easily. Take for instance the example in the expert review (Figure 3.8B): the AI developer notices that the “subtopic” angle category “water” has a different color pattern from other categories within the same category of the “topic” angle. He discovers that the solver achieves better accuracy only by chance. This advantage is also enabled by prismatic analysis: because prismatic analysis emphasizes multiple ways to categorize items and uses the same set of aggregation metrics, it facilitates comparison and contrast across angles. If one uses heterogeneous metrics to summarize each angle, comparison across angles is impossible.

6.4.3 *Sensemaking in interactive machine learning*

Sensemaking also happens in interactive machine learning (iML). In iML, AI developers interactively label, feature, and evaluate a model. Thus, the goal of sensemaking is to make sense of the data and the target concept and attempt to come up with features that express the concept in model languages. Strictly speaking, the featuring step is not part of the sensemaking process. However, as the process of featuring can impact foraging (i.e., the need to collect more data) and synthesizing (i.e., triggering modification of the schema), we include it in the modified notional model. The illustration in shown in Figure 6.4.

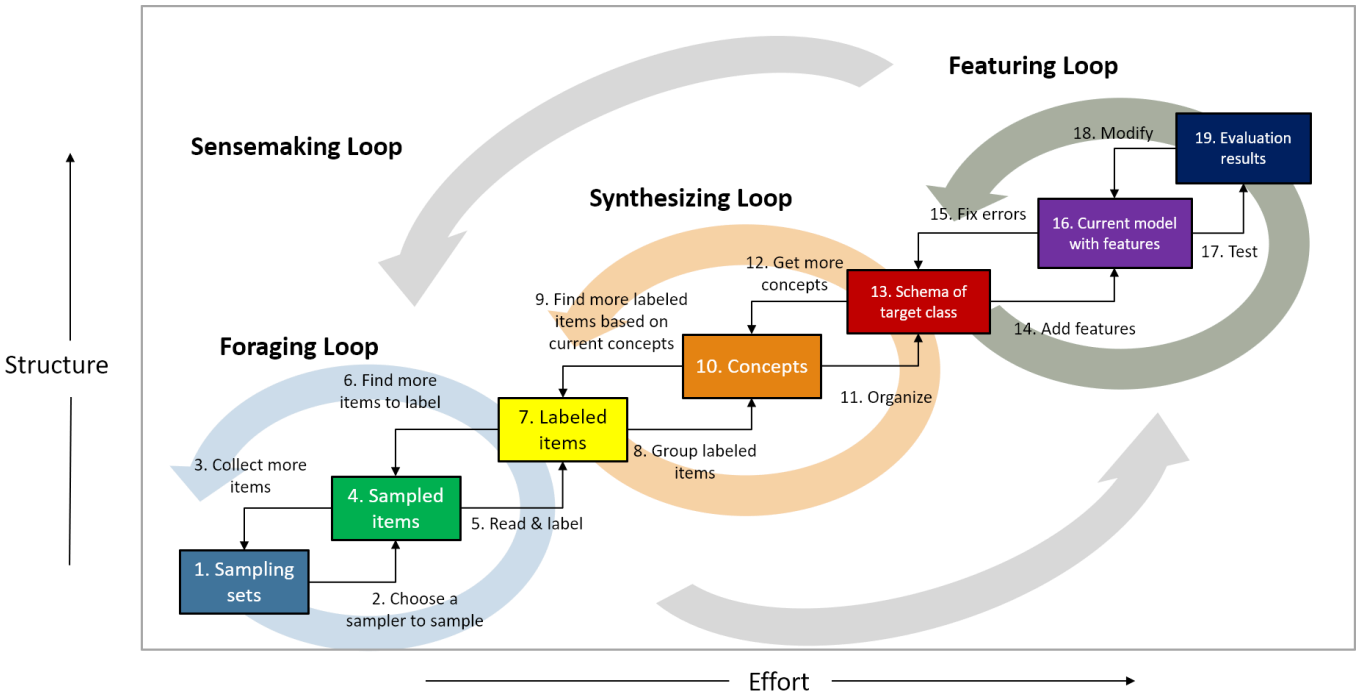

Figure 6.4: The iML sensemaking process (adapted from the sensemaking notional model)

The foraging loop of iML sensemaking focuses on the dataset. In the terms of iML, we call the original dataset **(1) sampling sets**. AI developers **(2) choose a sampler to sample** a set of **(4) sampled items** that they can **(5) read and label**. Once these items become **(7) labeled items**, AI developers may decide to move forward to the synthesizing loop or continue to **(6) find more items to label** or go even further to **(3) collect more items** from the sampling sets.

The synthesizing loop primarily focuses on coming up with an appropriate representation for the target class, which includes the **(10) concepts** and the **(13) schema of the target class**. To be clear, *schema* here refers to an organized set of concepts. For example, the target class "bird" can include multiple concepts: birds can fly, birds can swim, birds with long beaks, etc. However, to build the classifier, AI developers must further define a schema to include these

concepts, potentially creating a hierarchy. This is why both concepts and schema are included in this mapped model.

In the synthesizing loop, AI developers start by grouping **(7) labeled items** into concepts, and then **(11) organize** the concepts into the schema. From the schema, AI developers may recall another related concept, and thus seek to **(12) get more concepts**. Note that as the concept may not only be driven by data, but also by human knowledge, AI developers may come up with a set of concepts first, and then try to **(9) find more labeled items based on current concepts**. If they cannot find these based on existing labeled items, then they may go further to the foraging loop to search for items for the concept.

Given the schema, AI developers can start the featuring loop. The featuring loop is essentially a loop to determine an appropriate representation in the model language, whereas the synthesizing loop focuses on a representation in human language. AI developers must **(14) add features** to translate the schema to **(16) the current model with features**. In this process, AI developers may find errors in the schema, in which case they go back to **(15) fix errors** in the schema. Once the features are added into the model, AI developers **(17) test** the model to collect **(19) evaluation results**, and **(18) modify** the model (e.g., by adding more features, getting more labeled items) based on the results. There is also a possible sensemaking process from the evaluation result. However, compared with traditional AI development, changes in iML are relatively small and iterative. Therefore, sensemaking in the evaluation results in iML seems to be on a smaller scale. As a result, we do not emphasize it in this mapped model.

AnchorViz supports many of the aforementioned sensemaking steps. For example, for the foraging loop, users can examine a data point on the interface and drag it into the outer circle to

create an anchor. This is exactly the process of foraging for data, reading it, and creating a corresponding concept. With the created concept, users can come back to forage for more points that are related or in contrast to the created concept. This is also the process of sampling data from the sampling set.

When users have more anchors and attempt to place anchors closer to each other (e.g., a positive anchor on one side and a negative anchor on the other side), users are actually creating a schema for the target concept, which corresponds to the synthesizing loop. By interactively creating concepts and constructing the schema, and coming back to forage for more data, AnchorViz supports both the foraging and synthesizing loops in iML sensemaking. Future work can focus on supporting the featuring loop to close the whole sensemaking loop.

6.4.4 *Sensemaking in AI development*

In addition to evaluation analysis, AI development itself can also be considered a sensemaking process. In this case, AI developers are trying to make sense of the possible design choices, data, and configurations, and come up with the right combination of these for their target tasks. They must also make sense of possible ways to evaluate and measure the behavior of the AI systems. The mapping to Pirolli and Card's notional model is illustrated in Figure 6.5.

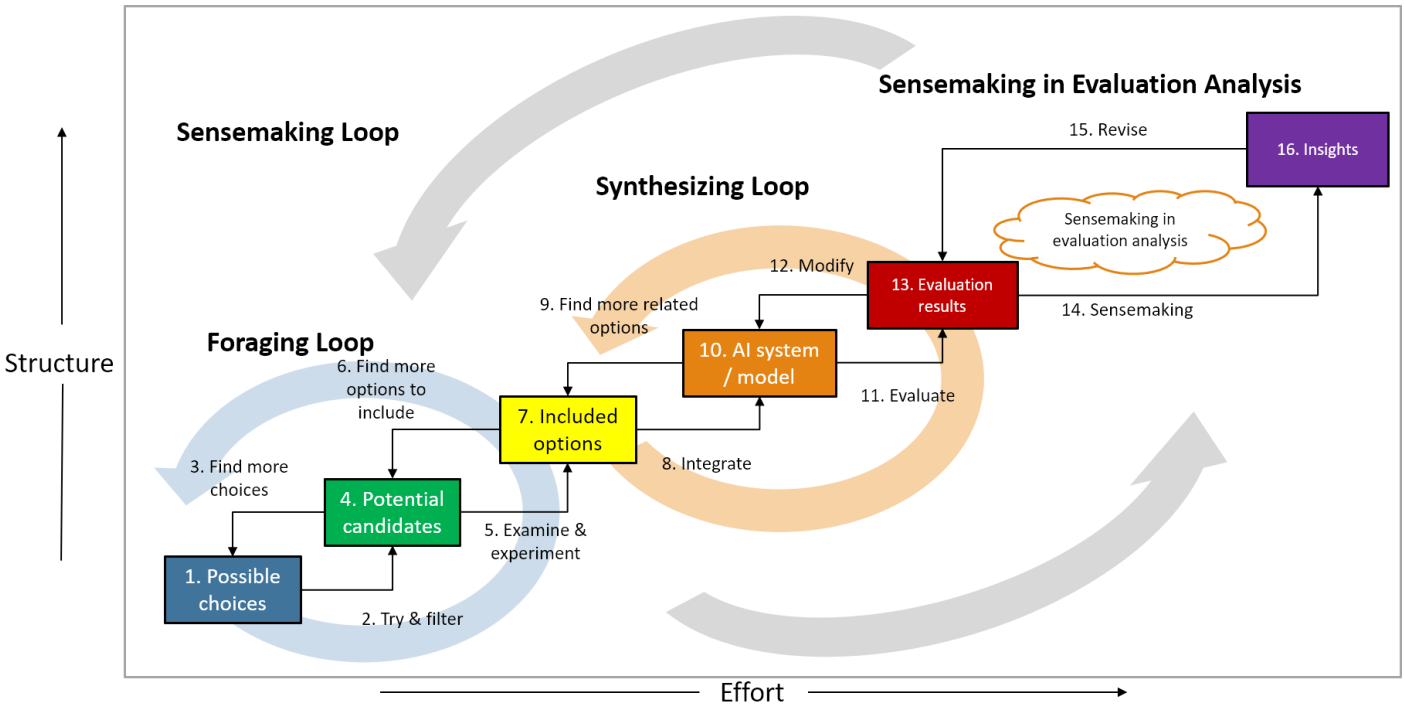

Figure 6.5: Sensemaking process in AI development workflows modified from Pirolli and Card's sensemaking notional model

In this modified version, the foraging loop concerns choices such as design variations, data, and parameters. In the world of AI, many datasets and algorithms are available, and many new datasets and algorithms are being collected and invented. However, not all of these are relevant to every AI problem. For example, some methods are more useful in image recognition, whereas other methods are specific to natural language processing. When it comes to building an image classifier, if an image dataset contains no images of the target class, it is not useful to include it. Even given the same model, some parameters that are not important for one problem (e.g., batch size) may be important for another problem. Thus, AI developers must **(2) try and filter** the **(1) possible choices** to determine a set of **(4) potential candidates**. Then they **(5) examine and experiment** with these candidates to determine whether they should use these candidates in their

systems, which becomes **(7) included options**. With a selected option, they may try to **(6) find more options to include**. If none of the candidates works, they return to **(3) find more choices**, which closes the *foraging loop*.

With the included options, AI developers must determine what the best ways are to **(8) integrate** the options into their **(10) AI system/model**. Adding a new component can lead to the need to **(9) find more related options**. For example, after adding a new feature, the model will require more labeled data for training. The next step is to **(11) evaluate** the new design/data/configurations. Then, another **(14) sensemaking** process occurs on the **(12) evaluation results**, as described in the previous subsection. The key outcome from the sensemaking process is the **(16) insights**, which can lead to either a **(15) revision** of the evaluation approach or further **(12) modification** of the AI system/model. These steps are a synthesizing loop combined with the evaluation analysis sensemaking.

6.4.5 *Directions for future work to use sensemaking theory to inform VA design for AI development*

Since the three modified notional models are only at a conceptual level, future work can focus on extending and modifying the initial models we have proposed above and come up with design considerations based on these models. In addition, future work can conduct a set of experiments to examine whether specific functionalities designed in existing tools support the sensemaking tasks well. Furthermore, more field studies are required to focus on how sensemaking happens in the AI development workflows. If future work can achieve the above three primary goals, the three modified notional diagrams will be useful to help examine and

discuss any of the tool designs for supporting evaluation analysis, interactive machine learning, and AI development workflows in general.

6.5 Concluding Reflections and Recommendations

Over the course of this dissertation, we have often been struck with how difficult it is to bridge the fields of HCI and AI. In the past few years, AI research communities and HCI communities have both sought to promote human-centered machine learning. For example, top conferences in HCI such as the ACM Conference on Human Factors in Computing Systems (CHI) or the ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW) usually include workshops on human-centered machine learning or human-centered data science. Top conferences in ML such as the International Conference on Machine Learning (ICML) or the Annual Conference on Neural Information Processing Systems (NeurIPS, formerly NIPS) also have workshops about interpretable or explainable machine learning. However, the two communities rarely interact.

One key reason is that the two communities value research very differently, and the languages the two sides speak are significantly different. It is hard to get an HCI paper published in an AI conference, and vice-versa. The pace of AI research is also faster than that of HCI research. Many AI papers now publish directly on arXiv[13] before any peer review, which forces AI researchers to daily chase after the latest publications. In contrast, HCI research takes time and effort to understand the target users and iterate the design. As an HCI researcher, it is not easy to communicate the value of HCI research to an AI researcher whose focuses are algorithm

[13] arXiv https://arxiv.org/

design. In addition, to design tools for AI researchers requires a certain level of understanding of AI systems, which HCI researchers do not often have.

In the past three years, we have been fortunate to have the opportunity to work with many outstanding AI researchers and developers (in our thesis, we refer to both as AI developers) for this thesis work. With their support, we have been able to pick up enough of their language to design tools for them. However, in the beginning connecting the two fields was not an easy task. It took time for us to explain how HCI research works, and to show how using an HCD approach to design a tool for them would make a difference. Although this thesis has turned out to be one of the connecting pieces for the two fields, we want to note that we did experience many struggles throughout the process. Therefore, we want to close the thesis with some recommendations for future researchers or designers who are interested in designing tools for AI development and bridging the gap between AI and HCI.

For HCI researchers and designers, it is important to understand that AI is a quickly evolving field. In the past three years, we witnessed how quickly the AI developers change their directions, tools, and development environments. Therefore, it is important to understand their needs in depth and find essential requirements that can carry over. For example, regardless of what directions AI developers focus on, they need ways to analyze their evaluation results. Although we do not claim that QSAnglyzer can be extended to a general-purpose tool, the design ideas in the tool, such as the visualization design for prismatic analysis, can be reused.

In addition, as we mentioned previously, picking up the language used in the AI field is critical to help design tools for AI developers. From our own experience, collaborating with AI developers and playing an active role to explain HCD methods is useful. It takes time, but the

efforts pay off. After building trust and understanding, AI developers are more willing to explain what they are doing in detail. Furthermore, sometimes people are not convinced as to the effectiveness of HCD, but once they find the design can solve their problems, they start believing in HCD methods. Thus, having a working prototype to demonstrate the design ideas is definitely valuable. We are very lucky that we can implement our own design ideas, which is not the case for every HCI researcher and designer. In such a case, we recommend teaming up with other people who have the requisite skill sets or hiring developers to implement the prototypes.

For AI developers who are interested in developing tools to support AI development, we recommend spending time learning HCD methods. Humans are complex and their behavior is situated in context. Just because a tool has powerful features does not necessarily mean it is a useful tool. Without an understanding of people's needs and natural workflows, powerful features can be useless. This is the lesson we learned that drove us to the road of HCI and HCD methods.

In addition, qualitative methods such as ethnographic studies provide in-depth illustrations about humans and their thoughts and behavior, which tells us why certain things matter. This is not what quantitative methods provide. We bring up this point directly, as in the AI field, quantitative metrics are far more important than qualitative descriptions. When we first came to the field of HCI with a bachelor's degree in computer science, we did not appreciate qualitative methods either. However, now we can say that our thesis work would not have succeeded without qualitative methods, especially the ethnographic studies and interviews we conducted. Therefore, for anyone who comes from the AI field, we highly recommend learning HCI and HCD methods. Humans are neither machines nor models; humans are *people*.

We consider our thesis work a small step toward bridging the AI and HCI fields. We strongly encourage future work to continue this effort. At the end of the day, AI systems are developed to be part of people's lives. When we design tools to support AI development, we are also bringing the perspectives of HCI into the AI world, such as an awareness of bias, and increasing the interaction between AI developers and HCI researchers and designers. It is our deep hope that one day, we can live in a world where everyone can utilize AI for their personal use, increasing the quality of our life. AI should be our friend, rather than systems that treat people as merely data points, dehumanizing the world.

BIBLIOGRAPHY

[1] Ahn, J.-W. and Brusilovsky, P. Adaptive visualization of search results: Bringing user models to visual analytics. *Information Visualization*, *8*, 3 (2009), 167–179.

[2] Alsallakh, B., Hanbury, A., Hauser, H., Miksch, S. and Rauber, A. Visual methods for analyzing probabilistic classification data. *IEEE Transactions on Visualization and Computer Graphics*, *20*, 12 (2014), 1703–1712.

[3] Alsallakh, B., Jourabloo, A., Ye, M., Liu, X. and Ren, L. Do convolutional neural networks learn class hierarchy? *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 152–162.

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

[5] Amershi, S., Chickering, M., Drucker, S. M., Lee, B., Simard, P. and Suh, J. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proc. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM (2015), 337–346.

[6] Amershi, S., Fogarty, J., Kapoor, A. and Tan, D. Effective end-user interaction with machine learning. In *Proc. Twenty-Fifth AAAI Conference on Artificial Intelligence* (2011).

[7] Amershi, S., Fogarty, J. and Weld, D. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Proc. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2012), 21–30.

[8] Ankerst, M., Elsen, C., Ester, M. and Kriegel, H.-P. Visual classification: an interactive approach to decision tree construction. In *Proc. KDD* (1999), 392–396.

[9] Atkins, D. *Revolutionizing Science and Engineering through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*. 2003,

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

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

[12] Baum, E. B. and Lang, K. Query learning can work poorly when a human oracle is used. In *Proc. International Joint Conference on Neural Networks* (1992), 8.

[13] Baxter, G. and Sommerville, I. Socio-technical systems: From design methods to systems engineering. *Interacting with Computers*, *23*, 1 (2011), 4-17.

[14] Bell, L. Artificial intelligence is now Intel's major focus, 2016. Retrieved from http://www.alphr.com/the-future/1004845/artificial-intelligence-is-now-intels-major-focus.

[15] Blei, D. M., Ng, A. Y. and Jordan, M. I. Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, Jan (2003), 993–1022.

[16] Braun, V. and Clarke, V. Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*, 2 (2006), 77–101.

[17] Brill, E., Dumais, S. and Banko, M. An analysis of the AskMSR question-answering system. In *Proc. Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing—Volume 10*, Association for Computational Linguistics (2002), 257–264.

[18] Brooks, M., Amershi, S., Lee, B., Drucker, S. M., Kapoor, A. and Simard, P. FeatureInsight: Visual support for error-driven feature ideation in text classification. In *Proc. 2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2015), 105–112.

[19] Brown, E. T., Liu, J., Brodley, C. E. and Chang, R. Dis-function: Learning distance functions interactively. In *Proc. 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2012), 83–92.

[20] Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L. and Blei, D. M. Reading tea leaves: How humans interpret topic models. In *Proc. Advances in Neural Information Processing Systems* (2009), 288–296.

[21] Chen, N.-C. and Kim, B. QSAnglyzer: Visual Analytics for Prismatic Analysis of Question Answering System Evaluations. In *Proc. 2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2017), 48–58.

[22] Chen, N.-C., Poon, S., Ramakrishnan, L. and Aragon, C. R. Considering time in designing large-scale systems for scientific computing. In *Proc. Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, ACM (2016), 1535–1547.

[23] Chen, N.-C., Ramakrishnan, L., Poon, S. S. and Aragon, C. Harnessing Complexity in High Performance Computing Ecosystems: A Complex Adaptive Systems Framework. In *Proc. Proceedings of the 52nd Hawaii International Conference on System Sciences* (2019).

[24] Chen, N.-C., Suh, J., Verwey, J., Ramos, G., Drucker, S. and Simard, P. AnchorViz: Facilitating classifier error discovery through interactive semantic data exploration. In *Proc. 23rd International Conference on Intelligent User Interfaces*, ACM (2018), 269–280.

[25] Chibana, N. 15 Stunning Data Visualizations (And What You Can Learn From Them), 2017. Retrieved from https://visme.co/blog/examples-data-visualizations/.

[26] Choi, M., Park, C., Yang, S., Kim, Y., Choo, J. and Hong, S. R. AILA: Attentive Interactive Labeling Assistant for Document Classification through Attention-Based Deep Neural Networks. In *Proc. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM (2019), 230.

[27] Clark, P., Etzioni, O., Khot, T., Sabharwal, A., Tafjord, O., Turney, P. and Khashabi, D. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proc. Thirtieth AAAI Conference on Artificial Intelligence* (2016).

[28] Collins, A. M. and Quillian, M. R. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, *8*, 2 (1969), 240–247.

[29] Cook, K. A. and Thomas, J. J. *Illuminating the path: The research and development agenda for visual analytics*. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2005,

[30] Dann, W. P., Cooper, S. and Pausch, R. *Learning to program with Alice*. Prentice Hall Press, 2008.

[31] DARPA, D. A. R. P. A. Explainable Artificial Intelligence (XAI), 2016. Retrieved from http://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf.

[32] Dewey, R. A. Gestalt Psychology, 2018. Retrieved from https://www.psywww.com/intropsych/ch04-senses/gestalt-psychology.html.

[33] Dhillon, I. S. and Modha, D. S. *Concept decomposition using clustering*. Google Patents, 2003.

[34] Dhillon, I. S. and Modha, D. S. Concept decompositions for large sparse text data using clustering. *Machine Learning*, *42*, 1-2 (2001), 143–175.

[35] Diehl, S. *Software visualization: Visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media, 2007.

[36] Dingen, D., van't Veer, M., Houthuizen, P., Mestrom, E. H., Korsten, E. H., Bouwman, A. R. and Van Wijk, J. RegressionExplorer: Interactive exploration of logistic regression models with subgroup analysis. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 246–255.

[37] Dobsa, J. and Basic, B. Concept decomposition by fuzzy k-means algorithm. In *Proc. Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, IEEE (2003), 684–688.

[38] Dowling, M., Wenskovitch, J., Fry, J., House, L. and North, C. Sirius: Dual, symmetric, interactive dimension reductions. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 172–182.

[39] El-Assady, M., Sevastjanova, R., Sperrle, F., Keim, D. and Collins, C. Progressive learning of topic modeling parameters: A visual analytics framework. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 382–391.

[40] El-Assady, M., Sperrle, F., Deussen, O., Keim, D. and Collins, C. Visual Analytics for Topic Model Optimization based on User-Steerable Speculative Execution. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 374–384.

[41] Endert, A., Fiaux, P. and North, C. Semantic interaction for visual text analytics. In *Proc. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2012), 473–482.

[42] Fails, J. A. and Olsen Jr, D. R. Interactive machine learning. In *Proc. Proceedings of the 8th International Conference on Intelligent User Interfaces*, ACM (2003), 39–45.

[43] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E. and Prager, J. Building Watson: An overview of the DeepQA project. *AI Magazine*, *31*, 3 (2010), 59–79.

[44] Fogg, A. A History of Deep Learning, 2017. Retrieved from https://www.import.io/post/history-of-deep-learning/.

[45] Forsythe, D. E. The Construction of Work in Artificial Intelligence. *Science, Technology, & Human Values*, *18*, 4 (1993), 460–479.

[46] Guo, P. J. Online Python tutor: Embeddable web-based program visualization for CS education. In *Proc. Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ACM (2013), 579–584.

[47] Hoffman, P. E. *Table Visualizations: A Formal Model and Its Applications*. University of Massachusetts. Lowell, 1999.

[48] Hohman, F., Head, A., Caruana, R., DeLine, R. and Drucker, S. M. Gamut: A Design Probe to Understand How Data Scientists Understand Machine Learning Models. In *Proc. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM (2019), 1–13.

[49] Holstein, K., Vaughan, J. W., Daumé, H., III, Dudik, M. and Wallach, H. Improving Fairness in Machine Learning Systems: What Do Industry Practitioners Need? In *Proc. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM (2019), 1–16.

[50] Hutchins, E. How a cockpit remembers its speeds. *Cognitive Science*, *19*, 3 (1995), 265–288.

[51] ISO 9241-210: 2010. Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems. *International Standardization Organization (ISO). Switzerland* (2009).

[52] Jandot, C., Simard, P., Chickering, M., Grangier, D. and Suh, J. Interactive Semantic Featuring for Text Classification. *arXiv preprint arXiv:1606.07545* (2016).

[53] Jeong, D. H., Ziemkiewicz, C., Fisher, B., Ribarsky, W. and Chang, R. iPCA: An Interactive System for PCA-based Visual Analytics. In *Proc. Computer Graphics Forum*, Wiley Online Library (2009), 767–774.

[54] Jurafsky, D. and Martin, J. H. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., 2009.

[55] Kahng, M., Andrews, P. Y., Kalro, A. and Chau, D. H. P. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 88–97.

[56] Kandel, S., Paepcke, A., Hellerstein, J. M. and Heer, J. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, *18*, 12 (2012), 2917–2926.

[57] Kapoor, A., Lee, B., Tan, D. and Horvitz, E. Interactive optimization for steering machine classification. In *Proc. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), 1343–1352.

[58] Kapoor, A., Lee, B., Tan, D. and Horvitz, E. Performance and preferences: Interactive refinement of machine learning procedures. In *Proc. Twenty-Sixth AAAI Conference on Artificial Intelligence* (2012).

[59] Karasti, H., Baker, K. S. and Millerand, F. Infrastructure Time: Long-term Matters in Collaborative Development. *Computer Supported Cooperative Work (CSCW)*, *19*, 3 (August 01 2010), 377–415.

[60] Kim, B. *Interactive and interpretable machine learning models for human machine collaboration*. Massachusetts Institute of Technology, 2015.

[61] Klein, G., Phillips, J. K., Rall, E. L. and Peluso, D. A. A data-frame theory of sensemaking. In *Expertise Out of Context*, (2007), 118–160.

[62] Ko, J., Nyberg, E. and Si, L. A probabilistic graphical model for joint answer ranking in question answering. In *Proc. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM (2007), 343–350.

[63] Krause, J., Dasgupta, A., Swartz, J., Aphinyanaphongs, Y. and Bertini, E. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *Proc. 2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2017), 162–172.

[64] Kulesza, T., Amershi, S., Caruana, R., Fisher, D. and Charles, D. Structured labeling for facilitating concept evolution in machine learning. In *Proc. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2014), 3075–3084.

[65] Kushman, N., Artzi, Y., Zettlemoyer, L. and Barzilay, R. Learning to automatically solve algebra word problems. In *Proc. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2014), 271–281.

[66] Kwon, B. C., Eysenbach, B., Verma, J., Ng, K., De Filippi, C., Stewart, W. F. and Perer, A. Clustervision: Visual supervision of unsupervised clustering. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 142–151.

[67] Lakkaraju, H., Kamar, E., Caruana, R. and Horvitz, E. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. In *Proc. Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[68] Lee, C. P., Dourish, P. and Mark, G. The human infrastructure of cyberinfrastructure. In *Proc. Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, ACM (2006), 483–492.

[69] Lee, H., Kihm, J., Choo, J., Stasko, J. and Park, H. iVisClustering: An interactive visual document clustering via topic modeling. In *Proc. Computer Graphics Forum*, Wiley Online Library (2012), 1155–1164.

[70] Lewis-Kraus, G. The Great A.I. Awakening, 2016. Retrieved from https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html.

[71] Lewis, D. D. and Catlett, J. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, (1994), 148–156.

[72] Lewis, D. D. and Gale, W. A. A sequential algorithm for training text classifiers. In *Proc. SIGIR'94*, Springer (1994), 3–12.

[73] Li, Q., Njotoprawiro, K. S., Haleem, H., Chen, Q., Yi, C. and Ma, X. EmbeddingVis: A Visual Analytics Approach to Comparative Network Embedding Inspection. *arXiv preprint arXiv:1808.09074* (2018).

[74] Little, J. 5 Benefits of Visualization Tools, 2015. Retrieved from https://www.inetsoft.com/blog/5-benefits-of-data-visualization-tools/.

[75] Liu, M., Liu, S., Su, H., Cao, K. and Zhu, J. Analyzing the noise robustness of deep neural networks. *arXiv preprint arXiv:1810.03913* (2018).

[76] Liu, M., Shi, J., Li, Z., Li, C., Zhu, J. and Liu, S. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, *23*, 1 (2016), 91–100.

[77] Liu, S., Bremer, P.-T., Thiagarajan, J. J., Srikumar, V., Wang, B., Livnat, Y. and Pascucci, V. Visual exploration of semantic relationships in neural word embeddings. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 553–562.

[78] Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proc. Advances in Neural Information Processing Systems* (2017), 4765–4774.

[79] Malik, M. M., Heinzl, C. and Groeller, M. E. Comparative visualization for parameter studies of dataset series. *IEEE Transactions on Visualization and Computer Graphics*, *16*, 5 (2010), 829–840.

[80] Meek, C. A characterization of prediction errors. *arXiv preprint arXiv:1611.05955* (2016).

[81] Ming, Y., Cao, S., Zhang, R., Li, Z., Chen, Y., Song, Y. and Qu, H. Understanding hidden memories of recurrent neural networks. In *Proc. 2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2017), 13–24.

[82] Mühlbacher, T. and Piringer, H. A partition-based framework for building and validating regression models. *IEEE Transactions on Visualization and Computer Graphics*, *19*, 12 (2013), 1962–1971.

[83] Muller, E., Gunnemann, S., Farber, I. and Seidl, T. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Proc. 2012 IEEE 28th International Conference on Data Engineering*, IEEE (2012), 1207–1210.

[84] Muller, M., Lange, I., Wang, D., Piorkowski, D., Tsay, J., Liao, Q. V., Dugan, C. and Erickson, T. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proc. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM (2019), 1–15.

[85] Munzner, T. *Visualization Analysis and Design*. CRC Press, 2014.

[86] Nagarajan, P., Warnell, G. and Stone, P. The impact of nondeterminism on reproducibility in deep reinforcement learning. In *2nd Reproducibility in Machine Learning Workshop at ICML 2018* (2018).

[87] Newell, A. and Simon, H. A. *Human Problem Solving*. Prentice-Hall Englewood Cliffs, NJ, 1972.

[88] Nocke, T., Flechsig, M. and Böhm, U. Visual exploration and evaluation of climate-related simulation data. In *Proc. Proceedings of the 39th Conference on Winter Simulation: 40 years! The Best is yet to Come*, IEEE Press (2007), 703–711.

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

[90] Olsen, K. A., Williams, J. G., Sochats, K. M. and Hirtle, S. C. Ideation through visualization: the VIBE system. *Multimedia Review*, *3* (1992), 48–48.

[91] Orban, D., Keefe, D. F., Biswas, A., Ahrens, J. and Rogers, D. Drag and Track: A Direct Manipulation Interface for Contextualizing Data Instances within a Continuous Parameter Space. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 256–266.

[92] Pezzotti, N., Höllt, T., Van Gemert, J., Lelieveldt, B. P., Eisemann, E. and Vilanova, A. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 98–108.

[93] Pirolli, P. and Card, S. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proc. Proceedings of International Conference on Intelligence Analysis*, McLean, VA, USA (2005), 2–4.

[94] Pu, P. and Chen, L. Trust building with explanation interfaces. In *Proc. Proceedings of the 11th International Conference on Intelligent User Interfaces*, ACM (2006), 93–100.

[95] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S. and Sohl-Dickstein, J. Survey of expressivity in deep neural networks. *arXiv preprint arXiv:1611.08083* (2016).

[96] Reisberg, D. *The Oxford Handbook of Cognitive Psychology*. Oxford University Press, 2013.

[97] Ren, D., Amershi, S., Lee, B., Suh, J. and Williams, J. D. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, *23*, 1 (2016), 61–70.

[98] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. S. and Silverman, B. Scratch: Programming for all. *Commun. Acm*, *52*, 11 (2009), 60–67.

[99] Ribeiro, M. T., Singh, S. and Guestrin, C. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386* (2016).

[100] Roza, F. End-to-end learning, the (almost) every purpose ML method, 2019. Retrieved from https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4.

[101] Russell, S., Dewey, D. and Tegmark, M. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, *36*, 4 (2015), 105–114.

[102] Sacha, D., Kraus, M., Keim, D. A. and Chen, M. VIS4ML: An ontology for visual analytics assisted machine learning. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 385–395.

[103] Sacha, D., Sedlmair, M., Zhang, L., Lee, J. A., Weiskopf, D., North, S. and Keim, D. Human-centered machine learning through interactive visualization. In *Proc.*, ESANN (2016).

[104] Sandberg, M. DataViz History: The Ghost Map: Dr. John Snow, 2013. Retrieved from https://datavizblog.com/2013/04/03/dataviz-history-the-ghost-map-dr-john-snow/.

[105] Seo, M. J., Hajishirzi, H., Farhadi, A. and Etzioni, O. Diagram understanding in geometry questions. In *Proc. Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014).

[106] Settles, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *6*, 1 (2012), 1-114.

[107] Sharko, J., Grinstein, G. and Marx, K. A. Vectorized Radviz and its application to multiple cluster datasets. *IEEE Transactions on Visualization and Computer Graphics*, *14*, 6 (2008), 1444–1427.

[108] Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*, (2003), 364–371.

[109] Shneiderman, B. Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. *ACM Transactions on Graphics*, *11*, 1 (1992), 92–99.

[110] Simard, P. Y., Amershi, S., Chickering, D. M., Pelton, A. E., Ghorashi, S., Meek, C., Ramos, G., Suh, J., Verwey, J. and Wang, M. Machine teaching: A new paradigm for building machine learning systems. *arXiv preprint arXiv:1707.06742* (2017).

[111] Smilkov, D., Carter, S., Sculley, D., Viégas, F. B. and Wattenberg, M. Direct-manipulation visualization of deep networks. *arXiv preprint arXiv:1708.03788* (2017).

[112] Sokal, R. R. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, *38* (1958), 1409–1438.

[113] Soo Yi, J., Melton, R., Stasko, J. and Jacko, J. A. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information visualization*, *4*, 4 (2005), 239–256.

[114] Stasko, J. and Zhang, E. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proc. IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*, IEEE (2000), 57–65.

[115] Stone, P., Brooks, R., Brynjolfsson, E., Calo, R., Etzioni, O., Hager, G., Hirschberg, J., Kalyanakrishnan, S., Kamar, E. and Kraus, S. Artificial intelligence and life in 2030. *One Hundred Year Study on Artificial Intelligence: Report of the 2015–2016 Study Panel* (2016), 52.

[116] Strobelt, H., Gehrmann, S., Behrisch, M., Perer, A., Pfister, H. and Rush, A. M. Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 353–363.

[117] Strobelt, H., Gehrmann, S., Pfister, H. and Rush, A. M. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 667–676.

[118] Stumpf, S., Rajaram, V., Li, L., Wong, W.-K., Burnett, M., Dietterich, T., Sullivan, E. and Herlocker, J. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, *67*, 8 (2009), 639–662.

[119] Talbot, J., Lee, B., Kapoor, A. and Tan, D. S. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In *Proc. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2009), 1283–1292.

[120] Tatu, A., Maaß, F., Färber, I., Bertini, E., Schreck, T., Seidl, T. and Keim, D. Subspace search and visualization to make sense of alternative clusterings in high-dimensional data. In *Proc. 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2012), 63–72.

[121] Tulving, E. Episodic and semantic memory. *Organization of Memory*, *1* (1972), 381–403.

[122] Van Den Elzen, S. and van Wijk, J. J. Baobabview: Interactive construction and analysis of decision trees. In *Proc. 2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE (2011), 151–160.

[123] Voorhees, E. M. The TREC-8 question answering track report. In *Proc. Trec*, Citeseer (1999), 77–82.

[124] Wang, D., Boytsov, L., Araki, J., Patel, A., Gee, J., Liu, Z., Nyberg, E. and Mitamura, T. CMU Multiple-choice Question Answering System at NTCIR-11 QA-Lab. In *Proc. NTCIR* (2014).

[125] Wang, J., Gou, L., Shen, H.-W. and Yang, H. DQNViz: A visual analytics approach to understand deep Q-networks. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 288–298.

[126] Ware, C. *Information Visualization: Perception for Design*. Elsevier, 2012.

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

[128] Weick, K. E., Sutcliffe, K. M. and Obstfeld, D. Organizing and the process of sensemaking. *Organization science*, *16*, 4 (2005), 409–421.

[129] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F. and Wilson, J. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics* (2019).

[130] Williams, E. Experimental designs balanced for the estimation of residual effects of treatments. *Australian Journal of Chemistry*, *2*, 2 (1949), 149–168.

[131] Wilson, A. T. and Potter, K. C. Toward visual analysis of ensemble data sets. In *Proc. Proceedings of the 2009 Workshop on Ultrascale Visualization*, ACM (2009), 48–53.

[132] Wobbrock, J. O. and Kientz, J. A. Research contributions in human-computer interaction. *interactions*, *23*, 3 (2016), 38–44.

[133] Wolfinger, R. D., Tobias, R. D. and Sall, J. Mixed models: a future direction. In *Proc. Proceedings of the Sixteenth Annual SAS Users Group Conference, SAS Institute Inc., Cary, NC* (1991), 1380–1388.

[134] Wongsuphasawat, K., Smilkov, D., Wexler, J., Wilson, J., Mane, D., Fritz, D., Krishnan, D., Viégas, F. B. and Wattenberg, M. Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics*, *24*, 1 (2017), 1–12.

[135] Xu, C., Tao, D. and Xu, C. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634* (2013).

[136] Yang, Q., Suh, J., Chen, N.-C. and Ramos, G. Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. In *Proc. Proceedings of the 2018 Designing Interactive Systems Conference*, ACM (2018), 573–584.

[137] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. and Lipson, H. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[138] Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *Proc. European conference on computer vision*, Springer (2014), 818–833.

[139] Zhang, J., Wang, Y., Molino, P., Li, L. and Ebert, D. S. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 364–373.

[140] Zhao, J., Xie, X., Xu, X. and Sun, S. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, *38* (2017), 43–54.

[141] Zhao, X., Wu, Y., Lee, D. L. and Cui, W. iForest: Interpreting Random Forests via Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, *25*, 1 (2018), 407–416.

[142] Zheng, Z. AnswerBus question answering system. In *Proc. Proceedings of the Second International Conference on Human Language Technology Research*, Morgan Kaufmann Publishers Inc. (2002), 399–404.